3rd ASIAN CONFERENCE

ON

INTELLIGENT GAMES AND SIMULATION

GAME-ON'ASIA 2011

3rd ASIAN SIMULATION

TECHNOLOGY CONFERENCE

ASTEC'2011

EDITED BY

Chek Tien Tan

MARCH 1-3, 2011

Digipen Institute of Technology

SINGAPORE

A Publication of EUROSIS-ETI

3rd Asian Conference
on
Intelligent Games and Simulation

3rd Asian Simulation
Technology Conference

SINGAPORE

MARCH 1 - 3, 2011

Organised by

ETI

Sponsored by

EUROSIS

Digipen

Co-Sponsored by

Ghent University

University of Skovde

The Higher Technological Institute

Hosted by

Digipen Institute of Technology

Singapore

ASTEC EDITORS

General Conference Chair

Chek Tien Tan, Digipen Research Centre, Singapore

General Program Chair

Donny Soh, Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR)

Past Conference Chair

Wenji Mao, Chinese Academy of Sciences, Institute of Automation, Beijing, China

INTERNATIONAL PROGRAMME COMMITTEE

Simulation Methodology

Hamid Demmou, LAAS/CNRS, Toulouse Cedex France
Helge Hagenauer, Universitaet Salzburg, Salzburg, Austria
David Hill, Universite Blaise Pascal, Aubiere, France
Vladimir Janousek, Brno University of Technology, Brno, Czech Republic
Xiaolin Hu, Georgia State University, Atlanta, USA
Panajotis T. Katsaros, Aristotle University Thessaloniki, Thessaloniki, Greece
Henrikas Pranevicius, Kaunas University of Technology, Kaunas, Lithuania
Damien Trentesaux, LAMIH/SP, le mont Houy, Valenciennes cedex, France

Simulation Tools

Xiaochen Li, Institute of Automation, Chinese Academy of Sciences, China
Renate Sitte, Griffith University, Gold Coast, Australia
Alfonso Urquía, UNED, Madrid, Spain

Agent Based Simulation

Marina Bagic, University of Zagreb, Zagreb, Croatia
Andre Campos, Natal RN, Brazil
Joel Colloc, Universite Lumiere Lyon 2, Lyon, France
Jacinto Davila, Universidad de Los Andes. Merida. Venezuela
Julie Dugdale, Institut IMAG, Grenoble Cedex, France
Alfredo Garro, Universita' della Calabria, Arcavacata di Rende, (CS)Italy
Wenji Mao, Chinese Academy of Sciences, Beijing, P.R. China
Juan Merelo, Telefonica, Spain
Lars Moench, FernUniversitaet in Hagen, Hagen, Germany
Andreea Monnat, University of Luxembourg, Luxembourg
Michael J. North, Argonne National Laboratory, Argonne, USA
Eugenio Oliveira, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal
Isabel Praca, Instituto Superior de Engenharia do Porto, Porto, Portugal
Maria Joao Viamonte, Instituto Superior de Engenharia do Porto, Porto, Portugal
Bejrouz Zarei, Management College of Tehran University, Tehran, Iran

Aerospace Simulation

Nima Amanifard, University of Guilan, Rasht, Iran
Ernst Kesseler, NLR, Amsterdam, The Netherlands
Zdravko Terze, University of Zagreb, Zagreb, Croatia
Kai Virtanen, Helsinki University of Technology, HUT, Finland

Engineering Simulation

Chrissanthi Angeli, Technological Education Institute of Piraeus, Athens, Greece
Jacques Andre Astolfi, Brest Armess, France
Josko Dvornik, University of Split, Croatia
Rene Heinzl, TU Wien, Vienna, Austria
Feliz Teixeira, University of Porto, Porto, Portugal
Morched Zeghal, National Research Council Canada, Ottawa, Canada

INTERNATIONAL PROGRAMME COMMITTEE

Simulation in Transport

Martin Adelantado, ONERA-CT/DPRS/SAE, Toulouse cedex, France
Ingmar Andreasson, KTH Vehicle Dynamics, Stockholm, Sweden
Abs Dumbuya, TRL Limited, Wokingham, United Kingdom
Rahila Yazdani, Carrickfergus, Co Antrim, United Kingdom

Robotics Simulation

Lyuba Alboul, Sheffield Hallam University, Sheffield, United Kingdom
Jorge M. Barreto, Campeche, Florianopolis, SC, Brazil
Nihat Inanc, Yuzuncu Yil University, Van, Turkey
Andres Kecskemethy, University Duisburg-Essen, Duisburg, Germany
Martin Mellado, Polytechnic University of Valencia (UPV), Valencia, Spain
Bogdan Raducanu, Campus UAB, Bellaterra (Barcelona), Spain
Ewald von Puttkamer, Kaiserslautern, Germany

Simulation in Manufacturing

Haslina Arshad, Universiti Kebangsaan Malaysia, Selangor, Malaysia
Alexander Felfernig, University of Klagensfurt, Klagensfurt, Austria
Michel Gourgand, Universite Blaise Pascal - Clermont Ferrand II, Aubiere, France
Peter Lawrence, Australian Catholic University, Fitzroy, Australia
Tina Lee, National Institute of Standards and Technology
J. Macedo, Universite du Quebec a Montreal, Montreal, Canada
Claude Martinez, IUT de Nantes, Carquefou, France
Habtom Mebrahtu, Anglia Ruskin University, Chelmsford, United Kingdom
Henri Pierreval, IFMA, Aubiere Cedex, France
Mehmet Savsar, College of Engineering & Petroleum, Kuwait University
Renate Sitte, Griffith University, Gold Coast Mail Centre, Australia

Simulation in Electronics

Mohsen Bahrami, Amirkabir Univ.of Technology, Tehran, Iran
Clemens Heitzinger, Purdue University, West Lafayette, USA
Javier Marin, Universidad de Malaga, Malaga, Spain
Maurizio Palesi, University of Catania, Catania, Italy

Simulation in Telecommunications

Alexandre Caminada, Universite de Technologie de Belfort, Belfort Cedex, France
Domenico Giunta, European Space Agency, Noordwijk, The Netherlands
Celso Massaki Hirata, CTA-ITA-IEC, S.J.Campos - SP - Brazil
Andreas Mueller, Institute of Mechatronics, Chemnitz, Germany

Automotive Simulation

Reza Azedegan, Urmia, Iran
Ignacio Garcia Fernandez, Universidad de Valencia, Valencia, Spain
Aziz Naamane, LSIS-EPUM, Marseille, France
Guodong Shao, NIST, Gaithersburg, USA

Graphics Simulation

Carlos Luiz N. dos Santos, UFRJ/COPPE/PEC/LAMCE, RJ-Brazil
Anders Hast, University of Gavle, Gavle, Sweden
Sudhir Mudur, Concordia University, Montreal, Canada
Daniela M. Romano, University of Sheffield, Sheffield, United Kingdom
Alfonso Urquia, ETSI UNED, Madrid, Spain

Ecological Sustainable Development

Philippe Geril, ETI Bvba, Ostend, Belgium

GAME ON'ASIA 2011

ASTEC'2011

All articles published in these Proceedings have been peer reviewed

EUROSIS-ETI Publications are Thomson-Reuters and INSPEC referenced

For permission to publish a complete paper write EUROSIS, c/o Philippe Geril, ETI Executive Director, Greenbridge NV, Wetenschapspark 1, Plassendale 1, B-8400 Ostend, Belgium.

EUROSIS is a Division of ETI Bvba, The European Technology Institute, Torhoutsesteenweg 162, Box 4, B-8400 Ostend, Belgium

Printed in Belgium by Reproduct NV, Ghent, Belgium
Cover Design by Grafisch Bedrijf Lammaing, Ostend, Belgium

EUROSIS-ETI Publication

ISBN: 978-9077381-60-1
EAN: 978-9077381-60-1

Preface

Welcome to the third joint conference consisting of the annual Asian Conference on Simulation and AI in Games (GAMEON-ASIA), and the annual Asian Simulation Technology Conference (ASTEC). The aim of this joint conference is to consolidate work in the technical aspects of games and simulations with special focuses in artificial intelligence, physics and graphics.

As a collective effort, annual GAMEON conferences have been held internationally in various parts of the world, which includes countries in Europe, North-America, the Middle-east and Asia. GAMEON conferences represent a prominent avenue for academics and practitioners alike to present novel research work in the video games domain. ASTEC on the other hand provides a similar avenue but focuses on work in the industrial simulations domain.

This year, GAMEON-ASIA 2011 brings several cutting-edge research papers in game artificial intelligence, computer graphics, game methodology and game design. ASTEC 2011 also presents to you papers from the management and security domains. As usual, all accepted papers have undergone rigorous reviews from the International Program Committee. Also include in our program are several invited talks from both academia and the games industry. With their expertise in computer game development, computer vision and computer graphics, we believe that much insightful knowledge could be gained to aid your game and simulation projects.

On behalf of the entire organizing committee, we wish to thank every contributing author for their effort in their preparations to submit their work in this conference. Our heartfelt thanks also goes to all the members of the International Program Committee for their precious time devoted to the review process. We also wish to thank all the program chairs, session chairs, presenters and attendees. Special mention goes to the Executive Editor, Philippe Geril, who has always given his utmost effort in making sure every detail falls into place. Last but not least, our gratitude goes to the European Multidisciplinary Society for Modelling and Simulation Technology (EUROSIS), for making this conference possible.

Lastly, we wish all of you a fruitful conference and an enjoyable stay in the unique city of Singapore. Again, we thank you for your valuable time in supporting and participating in this conference.

Chek Tien Tan
GAMEON-ASIA 2011 General Conference Chair
and ASTEC 2011 General Conference Chair

CONTENTS

CONTENTS

GRAPHICS

SCIENTIFIC PROGRAMME

INVITED PRESENTATION

REAL TIME RENDERING OF AMORPHOUS EFFECTS

Golam Ashraf and Koh Kok Weng
Department of Computer Science
National University of Singapore
13 Computing Drive, Coomputing 1, Singapore
E-mail: gashraf@nus.edu.sg

KEYWORDS
Natural Phenomena, Amorphous Phenomena, Dust, Smoke, Fire, Real time Rendering, GPU Acceleration, Perlin Noise

ABSTRACT

We present a simple generic method for rendering amorphous effects like smoke, dust and fire in real time, by combining surface-shaded procedural primitives, fractal noise, texture manipulation and meta-particle systems. The framework allows seamless transition from one type of effect to another, and requires simple high-level primitive creation and management. This method also achieves attractive transformations of arbitrary solid objects into dust, smoke or fire. We demonstrate several applications of our procedural amorphous effects that strike a good performance-quality compromise compared to volume rendering and billboard techniques. Our system runs on commodity graphics cards supporting Shader Model 3.0 at 60-120 fps. The reasonable visual quality and economical performance make this framework attractive for next generation games and interactive media.

1. INTRODUCTION

Amorphous special effects like smoke, dust and fire are common immersion elements in current games and animated features. Crumbling ruins, blazing tanks or shady opium dens are just not complete without these amorphous effects. Two techniques are widely used to model these phenomena: volumetric models and overlapped billboards. Volumetric models capture fine details of lighting and shape but are usually too expensive for real time applications. Overlapped billboards yield real-time performance, but have severe view angle limitations due to their flat structure.

In this paper, we explore a viable third alternative, namely, overlapped surface-shaded procedural 3D primitives. Using meta-particle systems to manage a bunch of procedural geometry, as opposed to micro-managing huge numbers of particles, we prove that it is possible to create an illusion of irregular volume with animated local features. Though this idea has been around for over two decades, it has not been utilized in a *generic amorphous effects* framework. Unlike existing work in procedural shading, we retain a high level of user-control over the general appearance of the phenomena, through procedural texture manipulation, and parameterized geometry distortion. These two techniques dramatically extend the expressiveness of our procedural shading framework, especially when we compare it with existing billboard systems. All three effects in Fig. 1 were created with the same shader code, and slightly different rules governing the meta-particle systems comprising only 10-25 low-resolution poly-spheres.

Our interest here is not to model physically correct simulation. Instead we want to capture essential properties of amorphous phenomena that make the visual representation believable enough. These shape, color and animation properties can be expressed as raw textures, procedural functions, and animated parameters that drive a common underlying shading network. Our goal is to produce models that take up minimum CPU resources, render fast and yet look good in 3D virtual worlds. The proposed framework naturally supports GPU acceleration, and performs in real time on current commodity graphics cards at 60-120fps.

We believe this work will be valuable to special effects developers for games and interactive media, since it is much more economical than volumetric modeling, yields better expressiveness than billboards, and is easy to control/implement. As shown in Fig. 1, it is possible to recreate believable effects quite easily from a single doctored input image.

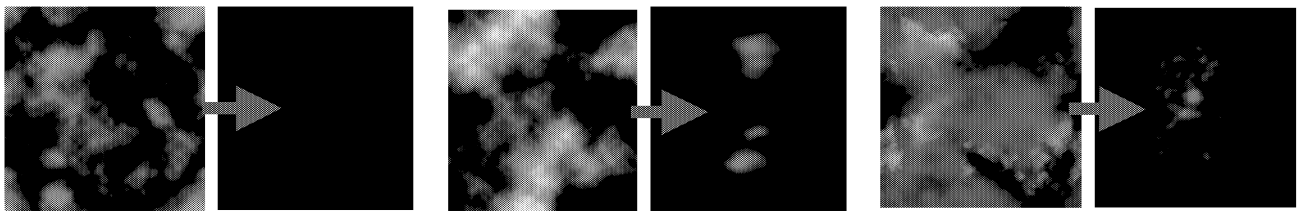


Figure 1. Real time dust, smoke and fire effects in 3D, created by modulating input textures (images on the left of arrows) with fractal noise before shading 10~25 overlapped spheres controlled by meta-particle systems

By sampling colors from real footage or procedural functions, it is possible to arrange a collage of expressive patterns into an input color texture, and then define temporal parameters that appropriately sample from this texture during the lifetime of the shaded 3D surface. The 3D surface can be a procedural primitive (e.g. sphere, cylinder, etc.) under the control of a meta-particle system. It could also be any other arbitrary shape (e.g. avatars, buildings, etc.) that is semantically remapped to the input texture space. Either way, it is easy to achieve transitions between solid and amorphous state, as well as change state between different amorphous forms. Furthermore, by combining the referenced color with multi-resolution noise, a large variety of results can be obtained from the same input texture. Lastly, the implemented framework incorporates a post processing layer, where image filters and color transforms can be applied to achieve the desirable final effect.

2. PREVIOUS WORK

Fluid modeling techniques can be broadly classified under two main categories, namely dynamics-based and ontogenetic-based approaches. We discuss trends in both these classes, as well as relevant work in hardware shading, before summarizing our contribution.

Dynamics-based approaches focus on accurate computation of low level particle physics and optical properties of fluids. Fluid animation using Navier Stokes Equation [Fay 1994; Foster and Metaxas 1996] and volumetric rendering [Blinn 1982; Antoine 2004] are popular examples from this class of methods. Apart from the computation intensiveness, a large number of particles are usually needed to represent small volumes, and thus often prove too heavy for real time application. Besides this, a third issue of control/predictability of simulation has challenged researchers. All three issues; i.e. memory, computation speed and control; are being incrementally alleviated. Jos Stam [1999] improved time step limitations in earlier techniques via a viscous flow algorithm where fluids are assumed to be incompressible and energy conserving. By adapting his algorithm to the GPU, [NVIDIA 2005] has opened up physics based simulation to a whole world of 2D interactive applications. However, 3D fluid flow continues to be too expensive for real-time applications. Klinger et. al [2006] report about 60~800 seconds of simulation and re-meshing time per frame of detailed interaction between fluids and obstacles, while McNamara et al. [2006] report about 95MB~600MB of memory usage per frame for 3D grids ranging from 30~50 units in each dimension. The final issue of simulation control by artists has been addressed eloquently by McNamara et al. [2006] by allowing a set of keyframes guide the simulation result, with performance speedup of several orders of magnitude. Better optimization techniques (e.g. adjoint method for more efficient gradient computation [McNamara et al. 2006]) and dimension reduction [Treuille et al. 2006] seem to be key directions forward.

Ontogenetic approaches place more attention on achieving visually convincing approximations, with economical memory and compute requirements. [Gardener 1985] proposed a fractal surface shading technique to achieve fuzzy edges for clouds. Ashraf and Wong [1999] created a believable pseudo-dynamics dust model for offline rendering, using a combination of Gardner's fractal shading and overlapped spheres. Our work is inspired by their approach, except that we choose procedural texture manipulation instead of procedural texture generation, in order to afford more user control over the final appearance. Furthermore, we achieve real time rendering with GPU accelerated geometry and pixel manipulations. Harris and Lastra [2001] achieved reasonable quality in real-time cloud rendering. They used light scattering pixel shading techniques, to render large superimposed billboards to form an illusion of clouds. Even though the visual appearance of billboard systems is good, the intra-particle animations are often static due to coarse texture mapping. Chen and Jim [1999] implemented a dense particle system for real-time dust generated by moving vehicles, but their method has the same limitations as the earlier discussed dynamics methods.

Rapid advances in programmable graphics hardware have encouraged significant acceleration of important rendering functions. Perlin's Noise [1985] function has been adapted for use in hardware shaders [Hart 2001; NVIDIA 2005]. Zdrojewska [2001] implemented turbulence enhanced fog in real-time. We draw inspiration from their noise superimposition technique, and adapt the idea to enhance color variation for the amorphous primitive surfaces.

We can draw a few conclusions from the existing state of the art. There is a lot of attention to amorphous phenomena models in the dynamics-based community. In comparison, there is hardly enough exploration in procedural models for such phenomena. Dynamics methods are still unsuitable for interactive applications. Thus, for example, the area of 3D amorphous phenomena is fairly undeveloped in games. We believe that almost all the animation, modeling and rendering features can be approximated ontogenetically. Our goal in this paper is to overcome the first two hurdles of real-time ontogenetic rendering: a) to make a bunch of overlapped primitives appear like an irregular volume that gradually diffuses away; b) to abstract parameters that capture important properties of amorphous phenomena.

We first cover a four-stage process, describing how we generate fractal alpha textures, and combine them with an input color texture to surface-shade overlapped 3D primitives. We then illustrate local-feature animation using texture and geometry distortion functions. We end the technical description with post-processing and effects compositing. After this we describe how to develop applications based on our rendering framework, covering important shader semantics, and examples on how to create specific kinds of amorphous special effects. Lastly we present performance statistics and wrap up with a qualitative analysis of our contributions.

3. IMPLEMENTATION

Let us describe the overall 4-stage render workflow as shown in Fig. 2. During the first stage we precompute a large matrix of Perlin's turbulence values [Perlin89]. We store the

values in two textures, one coarse-grain and the other fine-grain. Next during the texture preparation stage, a unique alpha texture is generated every timeframe by blending the precomputed coarse and fine grain noise textures. The drawing stage deforms the geometry representing amorphous volumes, and colors them with reference to the input color texture multiplied by the computed alpha texture. During the last stage, the drawn image is post-processed (e.g. blurring, color correction, etc.) and blended into the frame-buffer. In practice, the last 3 stages are separated for efficiency. If we need to render a cluster of surfaces with nearly the same properties, we could just reuse the blended noise texture from the texture preparation stage. Similarly, we could service the post-processing stage after all the amorphous surfaces in the view frustum have been rasterized and colored into a temporary buffer.

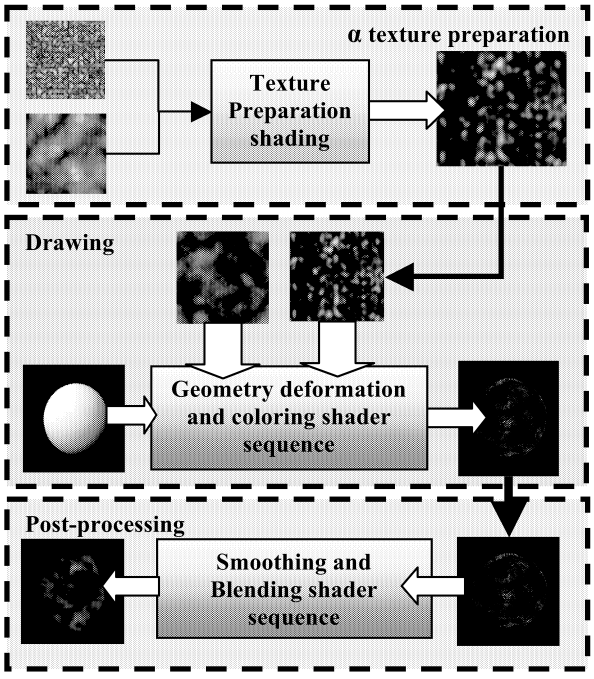


Figure 2: Amorphous Phenomena Rendering Overview

3.1 Noise Accumulation (Precomputation)

This stage serves to prepare a hardware friendly array of pre-computed noise values, and a convenient storage is a 2D texture. This is done before the starting of the rendering loop as a pre-computation step. We prepare 2 noise textures, using HLSL's *noise intrinsic* [MICROSOFT 2005], for the coarse and fine grain details for subsequent alpha modulation. The *noise intrinsic* is an implementation of Perlin's Noise, and runs on the CPU [MICROSOFT 2005]. As shown in the equations below, the procedure involves accumulating 256 iterations of a noise relation function into a color value for each of the Red, Green, Blue and Alpha channels, resulting in results similar to that in Fig. 3.

$$color_{red} = \sum_{i=0}^{S} \frac{|Noise(\eta_i[tex_x + 0, tex_y + 0])|}{i}$$

$$color_{green} = \sum_{i=0}^{S} \frac{|Noise(\eta_i[tex_x + 1, tex_y + 1])|}{i}$$

$$color_{blue} = \sum_{i=0}^{S} \frac{|Noise(\eta_i[tex_x + 2, tex_y + 2])|}{i}$$

$$color_{alpha} = \sum_{i=0}^{S} \frac{|Noise(\eta_i[tex_x + 3, tex_y + 3])|}{i}$$

In the equations above, ***tex*** represents the pixel-coordinate of the computed noise texture. For generation of coarse-grain and fine-grain detail, η is *2* and *15* respectively. *S* is set to be *256*.

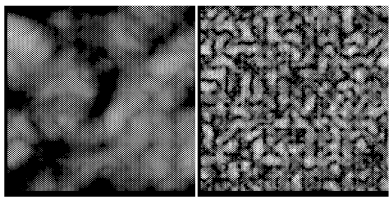


Figure 3: Coarse (η = 2) and Fine Grain (η = 15) Noise

The resolution of these noise maps are set to be 1024 X 1024 to reduce chances of texture repetition during shading.

3.2 Alpha Texture Preparation

In this stage, we need to prepare an alpha texture map to generate a distribution pattern for the shaded surface. Every pixel on this animated alpha texture is computed in a hardware pixel shader, as shown below.

First, we will need to determine the coordinates to sample to sample from the two noise textures. We choose a periodic function to inject subtle variation in sampling, and hence the alpha pattern itself, as follows:

$$tc_u = \cos(\frac{time \times speed}{100})$$

$$tc_v = \sin(\frac{time \times speed}{100})$$

where, *tc* is the texture coordinate offset.

The two pixels, P^f and P^c are then sampled from the fine-grain and coarse-grain noise textures respectively. The variable *tex*, once again denotes the coordinates of the alpha image that is being computed.

$$P^f = FineTexture(tex_{xy} + tc)$$

$$P^c = CoarseTexture(tex_{xy} + tc)$$

The raw alpha value α_{raw} is calculated as function of difference between the *alpha* channel of the coarse and fine noise pixels, and a sinusoidal offset driven by *rgb* values in the coarse noise pixel.

$$\phi_1 = \left(\sum_{i=0}^{2} P^c(i) \right) + tex_x + tex_y + time$$

$$\phi_2 = \left(\sum_{i=1}^{3} P^c(i) \right) + tex_x - tex_y + time$$

$$\omega_1 = \frac{\sin(Speed \times \phi_1)}{10}$$

$$\omega_2 = \frac{\cos(Speed \times \phi_2 \times 0.75)}{10}$$

$$\alpha_{raw} = 2 \cdot \left(P^f(4) - P^c(4) \right) + \omega_1 + \omega_2$$

$$\alpha_{smooth} = smooth_\vartheta(\alpha_{raw})$$

The raw alpha value is smoothed with a 2D Gaussian filter (see Appendix for details) before being stored in the alpha texture. Fig. 4 illustrates a time sequence of smoothed alpha textures generated by these calculations.

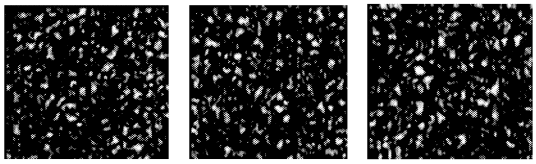


Figure 4: Animated Sequence of Alpha Textures

We believe that these noise-based alpha generation equations are an important contribution to the state of the art, as they facilitate a viable alternative to procedural fractal shading, championed by various existing ontogenetic shading frameworks [Gardner, 1985; Ebert et al. 1998; Ashraf and Wong, 1998]. It allows smooth modulation and animation of an input texture without noticeable aliasing or repetition. Fig. 5 illustrates this contribution as significantly different local features are achieved for a dust volume comprising 10 spheres, without distorting the geometry too much.

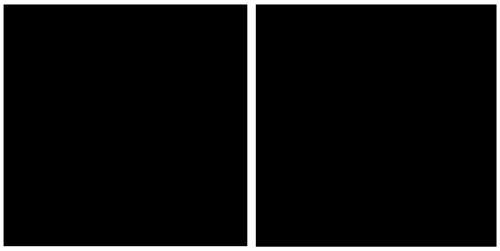


Figure 5: Alpha Textures achieving different local features

3.3 Geometry Deformation and Coloring

3.3.1 Vertex displacement

Local geometric distortions can be implemented on the CPU or the programmable vertex shader, following any periodic or force feedback based functions. As long as the vertices are well mapped into some kind of continuous logical space, a moderately scaled periodic distortion function should not yield any significant artifacts. The resulting distortion offset is added along the local vertex normals and tangents, so that results are invariant to object rotation. As a self-regulating measure against discernibly sharp edges, we scale down the alpha value proportionately to vertex displacement magnitude. Fig. 6 demonstrates how dramatic fire shapes are generated from a set of 25 low resolution poly-spheres, undergoing vertex distortion guided by 5 separate particle systems.

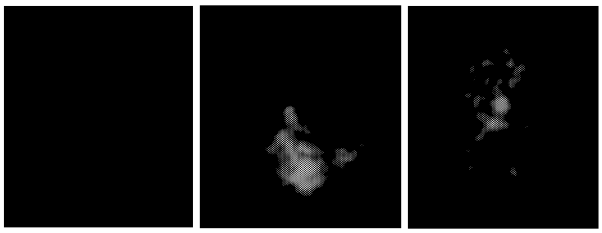


Figure 6: Local vertex distortion with alpha regulation

Regular primitives are generally quite easy to map. For arbitrary polygons, we re-map the vertices to a unit sphere, to extract the parameter driving the distortion function. Fig. 7 shows how different arbitrary geometry is shaded without any discontinuities.

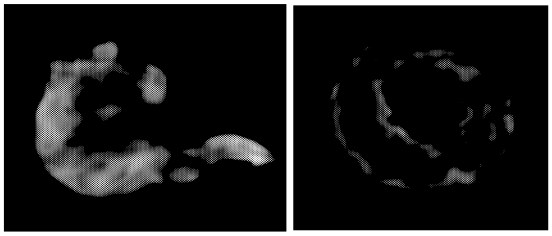


Figure 7: Smooth mapping for arbitrary geometry

We have extended the above idea to animated characters, as shown in Fig. 8. Using the same basic idea for smooth distortion and coloring, our framework takes over immediately after the hardware smooth skinning stage. It is easy to achieve simple transitions from one form to another, by blending between the input color textures. We are eager to extend these exciting results with simplified fluid dynamics in the near future.

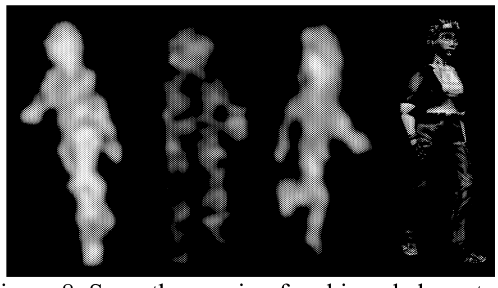


Figure 8: Smooth mapping for skinned characters

3.3.2 Texture Coordinate Manipulation

Animating vertex texture coordinates can create interesting sliding effects. Also, scaling of texture coordinates can yield different levels of granularity [Gardner 1985]. An example equation that combines both these factors is shown below:

$$uv' = uv + f_1(textureScale, time) \times \begin{bmatrix} f_2(u) \\ f_3(v) \end{bmatrix}$$

where *uv'* are the new texture coordinates, and $f_2(u)$ and $f_3(u)$ are some arbitrary mapping of model texture coordinates.

3.3.3 Lighting

We use per-vertex lighting with the Gouraud shading model, in favor of the more accurate (and expensive) per-pixel

Phong Shading model. Given our framework's expressive per-pixel color generation, we do not lose out much with interpolated light intensities. The lighting equation is slightly modified, so that the specular component of the shading is treated as residuals, which is useful when rendering dust and fire. Currently we do not add fog to enclosed volumes, but this could be aided with fast depth intersection accumulation.

3.3.4 Rasterization

Dust geometry is first drawn into an off-screen buffer with depth checking enabled to prevent overwriting occluding geometry. However, during the final alpha-blending with the frame-buffer, depth writing is disabled. The pixel shader simply combines the *alpha* value from the alpha texture computed in Sec. 3.2, and the *rgb* color value from the input color texture, using the vertex texture coordinates passed in by the rasterizer, in the following equations:

$$a_{rgba} = alphatex(v_{texcoord})$$

$$c_{rgba} = colortex(v_{texcoord})$$

$$p_{rgba} = \begin{bmatrix} c_{rgb} \times colorIntensity \\ a_a \times alphaMultiplier \end{bmatrix}$$

The output pixel p_{rgba} is written out to an off-screen buffer for subsequent post processing before final blending with the frame-buffer.

3.4 Post Processing

The contents of the off-screen buffer are smoothed (see Appendix) and color corrected based on the desired blurring and glow parameters, as shown below:

$$blurAmt = \|buffer(x,y)\| + 1$$

$$blurred(x,y) = \frac{1}{blurAmt}\left(smooth_{\vartheta}(buffer)\right)(x,y)$$

$$col_{rgb} = \begin{pmatrix} detailsAmt \times buffer(x,y)_{rgb} + \\ (1 - detailsAmt) \times blurred(x,y) + \\ glowAmt \times blurred(x,y)_a \end{pmatrix}$$

$$pp(x,y)_{rgba} = \begin{bmatrix} col_{rgb} \\ blurred(x,y)_a \end{bmatrix}$$

The processed pixel *pp(x,y)* is then alpha-blended into the frame-buffer as follows:

$$frame(x,y)' = \begin{matrix} pp(x,y) \times pp(x,y)_a \\ + \\ frame(x,y) \times (1 - pp(x,y)_a) \end{matrix}$$

4. Framework Application

Having presented the main technical ideas, we now discuss pertinent application issues like important shader parameters and a couple of possible modeling and animation scenarios to drive our ontogenetic rendering framework.

The designer needs to specify rules for generating the particle system, and the seed and rate of various animated parameters. Example rules could be conditions for placement of meta-particle (or 3D primitive) emitters; or shape transforms in the event of collisions, etc. Example parameters can be geometric attributes (e.g. affine transform and type of a primitive); or local shape, color and texture properties expressed via shader parameters.

We now explain some particle systems details implemented. Each emitter has an *age* scalar, a *position* vector, a *velocity* vector, and a *spray direction* vector. The *life* scalar grows with the application's *time* tick, and the emitter produces primitives if the *life* scalar is within its predetermined lifetime. The emission direction is affected by, but not equal to the *spray direction*. The variance is scaled by the emitter's *spread* property.

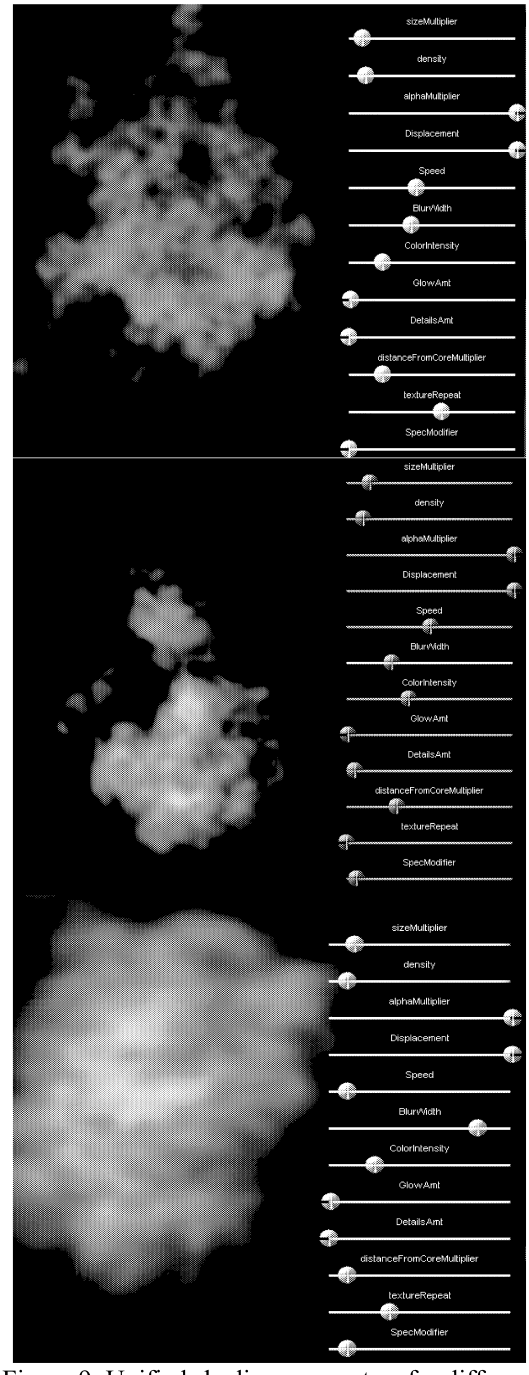


Figure 9: Unified shading parameters for different amorphous phenomena

Each meta-particle has a *position*, *direction*, *rotation*, and *size*. In turn, it also has a *p_age* scalar and *p_lifetime*. As *p_age* grows closer to its *p_lifetime*, the rendering parameters, *alpha* and *density*, in the shader are changed to

visually reflect its current age. All primitives are drawn using hardware instancing techniques [Microsoft 2005] to make rendering more efficient.

4.1 Shader Parameters

Fig. 9 demonstrates variations of a common set of parameters to achieve reasonably realistic results for three common amorphous phenomena, namely dust, fire and smoke. Each volume is made of typically 10~25 poly-spheres. Here is a description of important shader parameters and an example range of values that we cater for.

sizeMultiplier {0.0…3.0}: Controls the relative size of each dust primitive sent to this shader.

density {0.0…1.0}: Controls surface opacity. A density of 0.0 will cause the primitive to be invisible.

alphaMultiplier {0.0…1.0}: Global effect transparency. Used during scene blends.

displacement {0.0…6.0}: Affects how much a vertex can be displaced from its original position.

speed {0.0…10.0}: Scales internal animation time with respect to global time.

blurWidth {0.0…10.0}: Scales Gaussian blurring

colorIntensity {0.0…1.0}: Scales brightness of all rendered effect pixels.

glowAmt {0.0…1.0}: Scales brightness of blurred pixels.

detailsAmt {0.0…1.0}: Scales contribution from original un-blurred pixel.

distanceFromCoreMultiplier {1.0…8.0}: Scales vertex displacement to the transparency of the vertex. The bigger the number, the more transparent a vertex gets when it moves further away from the object center.

textureRepeat {1.0…8.0}: Controls the number of times the texture is wrapped around each dust primitive. The bigger this parameter is, the more times the texture is wrapped.

specModifier {0.0…1.0}: Specular scale. Affects shininess.

4.2 Trailing Dust/Smoke

This effect can model smoke trailing off a flying missile, or the dust raised from a speeding vehicle. To achieve such effects, we can use a single emitter with its *spray direction* opposing its *velocity*. Other parameters like *lifetime*, *growth*, etc. can be varied according to the desired end result. Fig. 10 shows a trail of sphere particles drawn with the shader configured to look like dust.

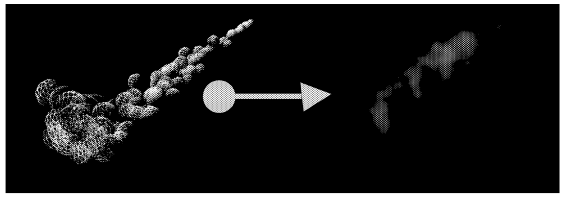


Figure 10: A spray emitting dust primitives

4.3 Amorphous Emission Shaped by Solid Geometry

This effect can model dust and smoke generated from collapsing buildings, game creatures morphing into dust and smoke after taking a hit, or objects engulfed in flames.

An object is often described as an array of polygons, which in turn are made up of vertices. We sample this array of vertices at discrete intervals. At the position of each sampled vertex, we attach an emitter to its position. The velocity of each emitter can then be affected by a gravity parameter, so that the amorphous volume created can then appear more realistic. The emitters can also be activated at different times for some time and before dying off. This can achieve more interesting non-uniform transition into amorphous volumes, as illustrated in Fig. 11.

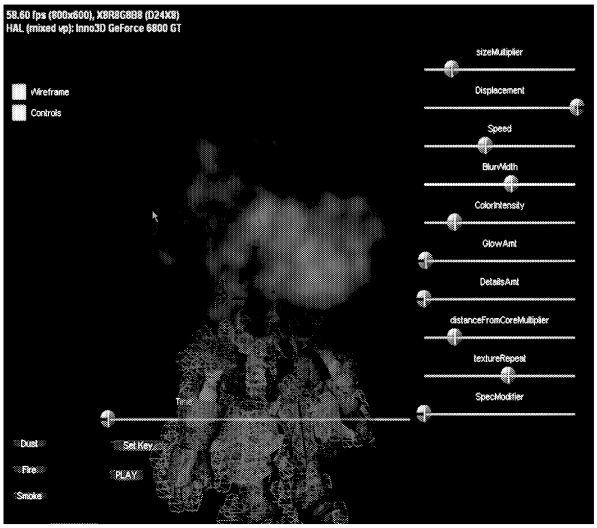


Figure 11: Particle emitters (red) activated at regular time intervals

5 PERFORMANCE ANALYSIS

We present three sets of render performance specifications, typically important to game and interactive media developers: a) number of meta-particles; b) number of emitters; c) screen resolution. We implemented our framework in DirectX9.0c using HLSL and C++, on an Intel Pentium4 3.06 GHZ with 1GB RAM, and an NVidia GeForce 6600 graphics card with 128MB DDR3 VRAM.

5.1 Impact of geometry

If we assume 15fps as the benchmark for real-time performance, each frame can take at most 66.67 milliseconds to render. Spheres of 10 latitudes and 10 longitudes were used; i.e. each sphere had 118 triangles. Care was taken to ensure that all the spheres were in the view frustum without any visibility culling.

Table 1: Number of spheres vs. rendering time

Number of spheres	Rendering time (ms)
1	5.12
5	5.85
10	6.95
30	10.87
50	14.29
100	25.00
200	43.48
300	68.97
500	98.04
1000	208.33

We can see from Table 1 that the rendering time is somewhat linearly related to the number of spheres. We could have at least 200 to 300 low-poly effect-primitives on screen at one time while still meeting the real-time requirement. Most of the figures presented here use less than 30 spheres, so it affords the user at least 10 such effects at a time, without any special optimization.

5.2 Impact of number of emitters

We used spray emitters that produced dust spheres at random intervals. On an average, each emitter produced 10 dust primitives every 3 seconds., as shown in Table 2.

Table 2: Number of emitters vs. rendering time

Number of emitters	Rendering time (ms)
1	7.33
5	11.11
10	17.64
15	23.81
20	28.41
25	31.25
30	33.20
35	38.46
40	43.48
50	55.56

We observed that the rendering time here is also somewhat linearly proportional to the number of emitters used. It goes on to show that simple particle systems scale well. Our rendering system could accommodate a maximum of 50 emitters on screen at one time before falling short of the 66.67ms benchmark.

5.3 Impact of Screen Resolution

Since the proposed framework relies heavily on the rasterization and pixel shading pipeline of the GPU, we need to investigate how the screen resolution impacts rendering speed. We conducted this test with an average of 80 spheres (each with 118 triangles) on screen at any one time, as shown in Table 3. We realized that beyond 640x480, the screen resolution significantly contributes to the rendering time. This is probably due to the fill-rate limitations of the GPU. We lose performance by about 30% at the highest resolution.

Table 3: Screen resolution vs. rendering time

	Rendering time (ms) per Frame		Average
	try 1	*try 2*	*try 3*
200X200	9.091	8.13	9.44
320X240	9.62	9.18	8.27
640X480	13.16	15.39	17.24
800X600	14.93	15.15	21.74
1024X768	18.87	20.41	17.55
1280X1024	21.28	25.00	26.32

In summary, our system yields reasonable performance on a modestly configured system. Since we use GPU acceleration, we can expect this performance to roughly double with each new generation of graphics cards. Since the performance is fairly predicable, with no hidden bottlenecks, etc., it affords developers a rough idea of design complexity, pretty much at the drawing board.

6. DISCUSSION AND FUTURE WORK

We have proposed a real-time rendering framework that can support ontogenetic modeling and animation of amorphous phenomena. Before we list our contributions, it is important to understand *why* our work is useful, and *what gaps* it has addressed.

Dynamics amorphous phenomena models generate great realism, but are hard to develop and control, and require massive computation and storage resources. They have not been yet able to provide answers for games and interactive media. Ontogenetic models seem to have more promise.

We have not proposed a dust model that works very well in racing dirt tracks. Neither is it a fire model that recreates great explosions from aerial bombs. Such models are probably generated and redeveloped many times over in production houses, where the target is to produce a compelling shot, or sequence, aided with very custom parameters and routines. This greatly reduces the *cross-applicability* of these models.

None of the elements that we use in this paper are entirely new. Procedural shaders, fractals, texture manipulation, overlapped primitives, particle systems… they are all well known, and individually well applied. It is the *generality* of ontogenetic modeling that is missing. Perhaps, it stems from a lack of belief in the possibility of simple and intuitive things solving challenging problems. Perhaps, it is hard to find commonalities between a dirt track and urban mayhem.

What we have done, is we have taken a small step to address this gap. We have demonstrated *three* different amorphous phenomena, namely: dust, smoke and fire, using the *same primitives*, *same particle systems* and the *same shader code*. We believe that this building block approach… reusing blocks interchangeably through a common set of strong semantic parameters… will truly encourage effects pipeline cross-applicability.

Let us now examine our tangible contributions. We have demonstrated a number of procedural texture and geometry manipulation techniques that can produce believable animation of local features in an amorphous volume. Paradoxically, this volume is an illusion. It is actually a set of procedurally shaded surfaces, rendered in a fraction of the time taken for true volumes. For the first time, we have demonstrated that overlapped primitives can be made to look like metaballs [Blinn, 1985] in real time. Users/designers can retain more control over the final appearance of the effect, by exercising their choice in color/pattern selection and semantic layout of the local features in the input texture. We used only one color texture for each effect, and easily doctored them via Photoshop's editing tools and filters. This contribution allows a more data driven flavor to special effects modeling, as opposed to procedure driven.

Our approach does have its limitations. When viewing the overlapped primitives at close proximity, or at high sharpness, the illusion of volume sometimes breaks down. An adaptive blurring scheme can help alleviate this somewhat. Also, the overall render quality still loses out to volume visualization. But we are comparing results generated in milliseconds (see Sec. 5) vs. in days [McNamara et al. 2004].

By creating images like Fig. 6 & 9, we hope to be able to excite more examination into high level ontogenetic animation and modeling. For example, how could we reuse the meta-particle render primitives for broad-phase collision detection and see the response in real time? This kind of serious work is being done in dynamics simulation [Treuille et al. 2006; Klinger et al. 2006], so we have reasons to be optimistic about ontogenetic modeling as well. The overall shape and animation of the simulated volume is quite dependent on the type of overlapped primitives. We hope to study different kinds of primitive modeling possibilities to address this aspect. We also hope to add dynamics attributes to the ontogenetic amorphous model. In particular, we are interested to extend the animated character application in Sec. 3.3.1 with some form of simplified dynamics. The shape emitter application in Sec. 4.3 could also be hooked up with fluid dynamics using the Lattice Boltzmann method [Chen and Doolean, 1998], and the calculation could also be accelerated on the GPU [Li et al. 2004]. In summary, we are pleased with the results of our generic real-time amorphous effects rendering framework, and excited about related ontogenetic modeling and animation research to follow.

REFERENCES

ASHRAF G. AND WONG K. C. 1999. Dust and Water Splashing Models for Hopping Figures. *Journal of Visualization and Computer Animation*, 10:4, pp. 193-213

BAKER D. AND BOYD C. 2001. Volumetric Rendering in Realtime. *www.gamasutra.com*

BLINN J. F. 1982. A Generalization of Algebraic Surface Drawing, *ACM Transactions on Graphics* 1(3), pp. 235–256.

CHEN J. X. AND FU X. AND WEGMAN E. J. 1999. Real-Time Simulation of Dust Behaviors Generated by. a Fast Traveling Vehicle, *ACM Transactions on Modeling and Computer Simulation, 9:2, pp 81–104*

CHEN S. AND DOOLEAN G. D. 1998. Lattice Boltzmann method for fluid flows. *Annu. Rev. Fluid Mech.*, 30:329–364.

EBERT D. S. AND MUSGRAVE K. AND PEACHEY D. AND PERLIN K. AND WORLEY S. 1998. Texturing and Modeling: A Procedural Approach, 2nd edition, ISBN 0-12-228730-4

FAY J. A. 1994. Introduction to Fluid Mechanics. *MIT Press*.

FOSTER N. AND METAXAS D. 1996. Realistic Animation of Liquids, *Graphics Interface '96,* pp 204-212

FORSYTH D. A. AND PONCE J. 2003. Computer Vision: A Modern Approach, *Prentice Hall.*

GARDNER G. Y. 1985. Proceedings of the 12th Annual Conference on Computer Graphics and interactive techniques, pp. 297 – 304

HART J. C. 2001. Perlin Noise Pixel Shaders, *Eurographics Workshop on Graphics Hardware*, pp.87-94.

HARRIS M. J. AND LASTRA A. 2001. Real Time Cloud Rendering*, Euro*graphics 2001, pp. 76-84.

KLINGNER B. M. AND FELDMAN B. A..AND CHENTANEZ N. AND O'BRIEN J. F. 2006. Fluid Animation with Dynamic Meshes, in *Proceedings of ACM SIGGRAPH 2006*, pp. 820-825.

MCNAMARA A. AND TREUILLE A. AND POPOVIC Z. AND STAM J. 2004. Fluid Control Using Adjoint Method, *ACM Transactions on Graphics (ACM SIGGRAPH 2004),* pp. 449-456

LI W. AND FAN Z. AND WEI X.AND KAUFMAN A.2004. GPU-Based Flow Simulation with Complex Boundaries*, Technical Report 031105*, Computer Science Department, State University of New York, Stony Brook

MICROSOFT 2005. DirectX 9 SDK (Oct. 2005). *Microsoft Corp.* http://mdsn.microsoft.com/directx

NVIDIA SDK 9.5 (2006). *NVidia Corporation* http://developer.nvidia.com/object/sdk_home.html

PERLIN K. 1985. An Image Synthesizer, ACM *SIGGRAPH*, pp. 287-296.

REEVES W. 1983. Particle Systems – A Technique for Modeling a Class of Fuzzy Objects, *ACM Transactions on Graphics*, 2:2, pp. 91-108.

STAM J. 1999. Stable fluids. Proceedings of *ACM SIGGRAPH '99*, pp. 121–128.

TREUILLE A. AND LEWIS A. AND POPOVIC Z. 2006. Model reduction for real-time fluids, *ACM Transactions on Graphics (ACM SIGGRAPH 2006),* pp. 826-834

TREUILLE A. AND MCNAMARA A. AND POPOVIC Z. AND STAM J. 2003. Keyframe Control of Smoke Simulations, *ACM Transactions on Graphics (ACM SIGGRAPH 2003),* pp. 716-723

WATT A. AND WATT M. 1992. Advanced Animation and Rendering Techniques: Theory and Practice. *Addison-Wesley.*

YNGVE, G.D. AND O'BRIEN, J. F. AND HODGINS, J.K. 2000. Animating Explosions, *SIGGRAPH 2000*, pp. 29-36

ZDROJEWSKA D. 2004. Real time rendering of heterogeneous fog based on the graphics hardware acceleration. *Central European Seminar on Computer Graphics for Students,* http://www.cescg.org/CESCG-2004/web/Zdrojewska-Dorota/

APPENDIX: TWO PASS GAUSSIAN CONVOLUTION

Forsyth and Ponce [2003] state that the double integration in 2D convolution can be simplified into 2 passes of 1D convolution for efficiency purpose if possible. This is especially useful Gaussian blur, which is separable.

We use the following kernel for convolution to achieve Gaussian blur:

$$Gauss_{1D} = \begin{bmatrix} 0.05 & 0.1 & 0.2 & 0.3 & 0.2 & 0.1 & 0.05 \end{bmatrix}$$

A 2D Gaussian kernel would be a matrix defined by: $Gauss_{2D} = Gauss_{1D} \times Gauss_{1D}^{T}$

We shall also use the following short-forms for the Gaussian kernels:

$$\vartheta(x) = Gauss_{1D}(x)$$

$$\vartheta(x,y) = Gauss_{2D}(x,y)$$

Let us denote the convolution of two images, I and W using the $*$ operator, such that

$$(I * W)(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (I(x-u, y-v) \times W(u,v)) \partial u \partial v$$

If the convolution is separable,

$$(I * W)'(x,y) = \int_{-\infty}^{\infty} (I(x-u,y) \times W(u)) \partial u$$

$$(I * W)(x,y) = \int_{-\infty}^{\infty} \left((I * W)'(x, y-v) \times W(v) \right) \partial v$$

Using the definitions above, we could define our 7-pixel Gaussian blur as:

$$(I * \vartheta)'(x,y) = \sum_{u=-3}^{3} \left(I(x - \frac{u}{w}, y) \times \vartheta(u) \right)$$

$$(I * \vartheta)(x,y) = \sum_{u=-3}^{3} \left(I(x, y - \frac{v}{h})' \times \vartheta(v) \right)$$

where w and h represent width and height of image I. The short form below represents the Gaussian blur operation on image I, using our 1D Gaussian kernel: $smooth_{\vartheta}(I) = (I * \vartheta)$

AUTHOR BIOGRAPHY

GOLAM ASHRAF received his PhD in Computer Engineering from NTU, Singapore. He teaches game development and animation at School of Computing, National University of Singapore. His research interests are in real time graphics, computational aesthetics, multimedia analysis, and pedagogical game design. Web-address: http://www.comp.nus.edu.sg/~ashraf/

KOH KOK WENG obtained his Bachelors degree in Computer Science from the National University of Singapore. He has deep interest in the application of novel computing techniques in visualization, security, data-processing.and business applications. He is currently working at Accenture Singapore.

GAME METHODOLOGY AND DESIGN

AUGMENTED REALITY GAMES: A REVIEW

Chek Tien Tan
Digipen Institute of Technology
PIXEL Building, #01-01
10 Central Exchange Green
Singapore 138649
chtan@digipen.edu

Donny Soh
Institute for Infocomm Research
1 Fusionopolis Way
#21-01 Connexis (South Tower)
Singapore 138632
cldsoh@i2r.a-star.edu.sg

ABSTRACT

This paper presents a review of the state of the art in Augmented Reality (AR) games. Distinguished advancements in terms of entertainment and serious games from both the research and industry are presented. These works are then analyzed across metrics like technology usage, game genre and chronology. Via this analysis, trends are extracted and novel insights into promising domains are eventually concluded, in both the perspectives of research and commercial development.

INTRODUCTION

We define Augmented Reality (AR) as a synthesized perspective of the real physical environment using computer-generated imagery. Additionally it is also commonly accepted to be interactive in real-time and registered in three dimensions (3D) [2].

Its potential can be seen in recent releases of the research AR applications such as the 6th Sense Technology by MIT Media Lab[25]. Still in research infancy, the inventors have agreed to release the source code which we expect will generate another wave of innovations in AR.

The interactivity and immersiveness of AR technologies make games one of its best applications. Given the huge and rapidly expanding game industry, its inherant ability to make new technology pervasive (like the graphics processing unit) and its enormous number of consumers [1], it is not surprising to see that a vast majority of work in both AR research and industry are focused in making games. Hence this paper aims to provide a review of the state of the art in the AR games domain.

To scope the domain, we only reference complete games that are 1. published research work or 2. commercial work that have been publicized on the internet. We also make the distinction between AR and Virtual Reality (VR). Easily confused, Virtual Reality (VR) performs somewhat the reverse of AR, placing the human into a completely virtual world, whereas AR is primarily the interaction of virtual objects into the real world (usually in as a video feed, gps data). This paper is about the latter and does not include VR games.

To the best of knowledge, there are no formal reviews done in the domain of AR games. There was however a paper that reviewed the AR domain in general, way back in 1997 [2]. In that paper, various applications of AR were presented but games were not in that list. Although a large number of work has been performed in AR games, a consolidation has not been done yet. There was also a case study review [18] that was performed by only describing five AR games separately. Without a thorough holistic view and analysis of current advancements, a lot of overlapping work will surface, especially in research.

Motivated by the lack of a holistic consolidated review of current AR work in the computer games domain, this paper aims to fill that cavity. The goal of this paper is to provide an initial framework for an analysis of the trends in current AR games such that researchers and practitioners are able to have a holistic view of the current state of the art in order to better advance AR game technology as well as make better commercial AR games. This paper by no means tries to cover the complete space of AR games but aims to cover the most high profiled ones.

The contributions in this paper includes

1. a review of state of the art AR games in research and industry, and

2. a trend analysis of these games.

The analysis results shown in this paper serves two purposes. Firstly, researchers will be able to target and work on areas that are currently lacking in terms of technology. Secondly, commercial game developers will be able to determine currently popular technologies to create or enhance their AR games. In terms of commercial viability, they can also use this review to select game genres that are currently lacking in AR games and get the first-movers advantage.

Games can be broadly classified into either entertainment games or serious games. Not surprisingly, AR games also fall into either of these categories. The structure of this paper will be to first describe key advancements in the entertainment games domain in Section , followed by the serious games domain in Section . Then

an analysis is provided in Section to cluster and identify common trends and pitfalls in current advancements, as well as provide directions for future AR work. Finally, a conclusion is given in Section .

ENTERTAINMENT GAMES

This paper defines entertainment games as computer games whose sole purpose is to provide entertainment to the player. Entertainment or player enjoyment is the original goal computer games were created for. The purpose of having such a definition is more obvious when compared to relatively newer domain of games known as serious games, which will be defined in Section .

This section lists, in reverse chronological order, some of the most prominent entertainment AR games created in both the research and commercial domains.

AR Defender

AR Defender [14] is a commercial casual game that was just recently released on the iPhone, a popular mobile smart phone product by Apple Inc. [13]. It makes use of its proprietary marker pattern printed on a card so that the software can make use of the phone's camera to detect the position and orientation to place a virtual tower. The goal of the game is to defend the tower by moving the camera and shooting various weapons at the enemy units that try to take down the tower. It is claimed to be the first complete and fun AR game on the iPhone [11].

Invizimals

Invizimals [24] is a commercial casual game that was released on the PlayStation Portable (PSP), a handheld gaming device by Sony Computer Entertainment [23]. It requires the player to lay proprietary printed marker patterns in the real-world. These markers are detected by the PSP's camera and rendered as traps in the virtual game world such that virtual animals can be hunted and captured by the players. It has been well received by consumers and received a top ranking in Amazon UK [12].

Art of Defense

Art of Defense [9] is a multi-player AR board game made for research purposes. It makes use of a classical physical board game placed on the tabletop but modified to contain multiple pre-defined marker patterns. Its software is built on a mobile smart phone which it makes use of to detect and track the marker. Virtual towers can then be built by players on these marker positions via the phone. They performed an empirical user study which has shown that players found it enjoyable although they also found it vastly different from traditional non-AR computer games.

AR Quotes

AR Quotes [14] is a commercial casual game that is also on the iPhone. It makes use of its proprietary marker pattern printed on a card so that the software can make use of the phone's camera to detect the position and orientation to place a virtual peg. The goal of the game is to toss virtual rings such that it falls on the peg. Although simplistic, it is the first game on the iPhone to use markers to overlay 3D content onto real-time video.

MyTown

MyTown [3] is a location-based commercial social game based on real-world property ownership released on the iPhone. The game simply makes use of the phone's Global Positioning System (GPS) to find out the player's location, then allow him/her to buy and own that real-world location virtually. The game also enables the player to drop virtual items based on proximity to these real-world locations. The AR technique used here does not actually superimpose the virtual graphics onto a video, but rather conceptually superimposes the virtual items onto the real world using the GPS. With their partnership with H&M and Travel Channel, MyTown served 14 million branded items in apps and drove thousands to their 200 retail stores across the country.

AR Squash

AR Squash [21] is an AR racquet sports game created for research purposes. It uses pre-defined markers, a charged-coupled device (CCD) camera connected to a personal computer, and a motion tracker. Unlike others, the markers in this game are not used to render objects on top of them. Instead, these markers are placed on walls and used to detect the 3D geometry of the real world. The ball is rendered virtually and the player hits the ball using a motion tracker mounted on a racket. The ball then moves realistically according to physics with the geometry estimated using the markers. This work shows an innovative and alternative use of marker-based technology.

Augmented Coliseum

Augmented Coliseum [20] is a mini RTS AR game created for research purposes. It uses pre-defined markers, cameras connected to personal computers and mini robots with movement controllers. The markers are placed on the mini robots such that virtual add-ons like weapons can be rendered on them. The player controls these robots with the movement controllers such that they will attack the other robots in the virtual world using the virtual weapons. Hence, they have demonstrated a deep fusion of both virtual (using weapons to attack) and real-world interactions (collisions) in this game.

CurBall

CurBall [17] is a casual curling-cum-bowling sports game created for research purposes. It uses pre-defined markers, cameras connected to personal computers and a motion detector. The markers are placed on easily movable physical objects which are rendered as obstacles on the screen. The motion detector is placed in a ball which is also rendered on the screen as a ball. The game is a two-player cooperative game. The first player controls the ball and the task is to get the ball to a certain goal position without hitting the virtual obstacles. The first player can see the entire game world with the rendered obstacles and ball. The second player can only see and move the physical obstacles. The two players are physically separated and are only virtually connected thru a network. Hence they have to cooperate and shift the obstacles while rolling the ball to reach the goal. They have hence presented a novel cooperative networked game.

Butterfly Effect

Butterfly Effect [26] is a casual AR game made for research purposes. It uses a head-mounted display (HMD) [2] and a customized rod controller. The aim of the game is to capture all the virtual butterflies that are rendered in front of the player. The rod serves as a device to attract the butterflies and make them move in a regular fashion such that they can be caught easily. Catching butterflies simply means moving the HMD close enough to them. The butterflies are rendered in a 3D volume around the player, so the player has to move and make use of real objects to get to these butterflies. Their game has shown to make good use of the 3D space around the player without using markers.

ARBattleCommander

ARBattleCommander [28] is an AR real-time strategy (RTS) game made for research purposes. It uses a video see-through HMD [2] to both display the virtual objects on the see-through real world as well as obtain inputs from the player via a heads-up display (HUD) menu in the HMD. It also uses a GPS to track position and a motion sensor to track motion. The system also detects a glove worn on the player's hands which allows him/her to issue commands on the HUD using physical hand movements. The player is the commander of a team of units who can then make use of his hands and the HUD to control them like in that of a standard RTS game. The limited field of view also acts like a natural fog-of-war present in RTS games.

ARQuake

ARQuake [5, 29] is an AR version of the popular first-person shooter (FPS) Quake game by id Software [10] made for research purposes. The system uses a video see-through HMD, a GPS, a hybrid magnetic and inertial orientation sensor, a custom made gun controller, and a standard laptop carried on a backpack. A full replica of the real world map is created as a Quake map. The software algorithm's task is hence to constantly perform accurate registrations of the virtual map with the real world. The player then plays the FPS by using the orientation sensitive HMD to aim, and pressing the gun controller to shoot. Though the game has never become commercial, ARQuake was the first fully working outdoors AR game. Hence it has generated substantial interest in the AR community back then.

SERIOUS GAMES

The term "serious games" [36] is generally accepted to mean games with a purpose, going beyond entertainment to deliver engaging learning experiences across a wide range of sectors [35]. Examples are educational games for teaching aids in schools, combat games for military simulations in the army, and advertising games for marketing purposes in commercial companies.

Similar to the previous section, this section lists, in reverse chronological order, some of the most prominent serious AR games across both the research and commercial domains.

Tangible Cubes

Tangible Cubes [15] is an educational AR game to educate children on endangered animals. It uses pre-defined markers that are viewed on a video HMD. It also allowed game administrators to view the augmented scene using web cameras mounted on personal computers and notebooks. The markers are actual numbers and symbols printed on cubes and the task of the child is to turn the cubes to indicate their choice when asked to find a certain animal. The respective animals are virtually rendered on the choice markers. One cube also overlays videos on it to describe the questions. Their results indicate that children found it more enjoyable and preferred playing the AR version rather than the traditional version of the computer game, although they did mention that the AR version is harder.

Learning Words

Learning Words [16] is an educational AR game to help children learn words. This game is done by the same researchers who created Tangible Cubes. It also uses HMDs, markers and cameras. In this game, the child is required to correctly spell words by placing and aligning the markers representing each alphabet in their correct positions. Several markers are also used as user interface menu buttons and video overlay hints. Similar to Tangible Cubes, their study also shows that children liked the AR version of this game more.

Shelf Stack
Shelf Stack [4] is an AR game used for upper-limb stroke rehabilitation. It uses pre-defined markers viewed through cameras on a personal computer. Markers are placed as the game designer wishes on the player's desktop, which then sits a virtual object on top of it in the screen. Many of these objects will be rings. At each stage of the game, one of the rings will be highlighted along with another object (like a teapot). The aim of the game is to keep placing the highlighted item onto the correctly highlighted ring. A time limit is imposed with a scoring mechanism.

AR Racing
AR Racing [22] is a driving AR game that teaches hand coordination, including to people with disabilities. It uses markers, HMDs, pinch gloves and the Wiimote, a motion sensing device by Nintendo [27]. The goal of the game is to move the virtual car around the scene using the Wiimote without colliding with certain virtual obstacles placed on marker positions in the real world. These virtual obstacles can be moved around using the glove. Their results show that multi-modal interaction games are beneficial in the domain of serious games.

Environmental Detectives
Educational Detectives [19] is multi-player educational AR game to support learning in late high school and early college environments. It simply uses a GPS on a smart phone to display the player's current position on a virtual map. The real environment is not shown on the smart phone. Similar to MyTown, this game conceptually superimposes the virtual items onto the real world using the GPS. The goal of the game is to find out the culprits, causes and solutions to a scientific sabotage case by moving to various real-world locations and conducting interviews with virtual persons as well as reading virtual documents. The cases are modeled such that the information to be obtained are related to scientific topics the student players are studying. They have provided case studies of their development process which serves as a good reference in terms of serious AR game design.

GenVirtual
GenVirtual [6] is a musical educational AR game to help people with learning disabilities. It uses pre-defined markers and a camera mounted on a personal computer. Each marker represents a certain musical note and they can be placed according to what the therapist intends to teach. Cube buttons are then rendered on each marker. A musical sequence is then demonstrated at the start of the game, with each button appearing to indicate the musical note correspondence. The goal of the game is to correctly repeat the sequence. The corresponding note is played when the player's hand is placed over the button (marker).

ARVe
ARVe (Augmented Reality applied to the Vegetal field) [30] is an educational AR game for rehabilitation of cognitive disabled children to make proper decisions. It uses pre-defined marker patterns printed on a picture book. The children views the book through a camera mounted on a personal computer, in which virtual plant entities are rendered on the markers. The goal of the game is to place the correct virtual plant entities (by shifting some moveable markers) in the same positions shown on a reference page. For example one task might be to place the flowers, seeds and leaves of pears and strawberries in their correct positions. Their research showed that cognitive disabled children were very enthusiastic and showed a higher motivation to complete the tasks than other children.

THE AR GAME DOMAIN

With a description of all the prominent work performed in AR games, this section aims to consolidate and analyse the current state of AR games. The analysis focuses on the spectrums of technology used and game genre implemented such that insights can be effectively drawn using these spectrums as a basis.

Technology
The current state of technology usage can be summarized in the graph shown in Figure 1. Note that technology is not divided into hardware and software because AR games are currently still primarily dependent on the choice of hardware. The software program is still very much built around the hardware that are chosen. Descriptions of each of the technologies on the y-axis are already given alongside the game descriptions in the previous section on Entertainment Games.
As shown in Figure 1, the work done in earlier years mainly make use of highly specialized hardware like HMDs, gloves and motion sensors. These hardware used are mostly costly lab equipment which is why these work primarily stayed in the lab without being commercialized. In recent years, especially in 2010, it can be seen that marker-driven detection and tracking through a simple web camera or phone camera has become immensely popular. The cost of equipment is simply a cheap camera and some paper to print the marker patterns. This easy accessibility to equipment meant that even hobbyist researchers without funding can investigate AR games, and that game companies can make AR games easily available to the masses. This also coincides with the rapid adoption of smart phones and ultra-portable laptops by consumers. It is noted however that there are still recent work using HMDs [22, 15, 16]. Hence HMDs are not completely obsolete especially when a high level of immersion is required.

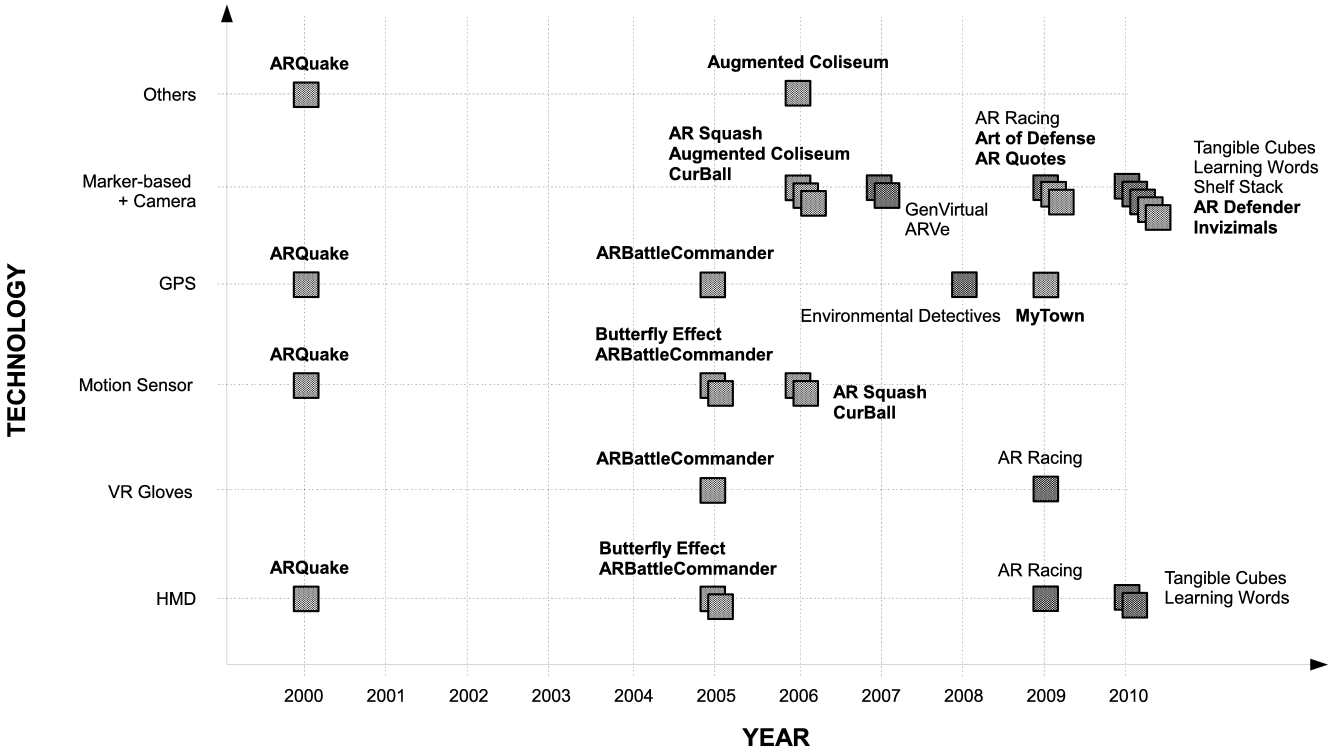


Figure 1: A graph of technology usage versus year of release/publication. Blue color denotes entertainment games while red color denotes serious games. It can be seen that in general, the trend is moving from multi-modal expensive hardware to simply using markers and cameras.

Game Genre

The current state of game genres implemented can be summarized in the graph shown in Figure 2. Most of the mainstream game genres are listed on the y-axis, namely First Person Shooters (FPS) [31], Real-Time Strategy (RTS) games [32], Role-Playing Games (RPG) [33], and Sports games [34].

Although there is no obvious trending in terms of chronology, at least it can still be seen from Figure 2 that more AR games are being created as time goes by. It is also seen that in general, AR games tend to deviate from the mainstream game genres. Undoubtedly, serious games exhibits this trait more obviously as current serious AR games are used mostly for education or medical rehabilitation. If serious AR games are to be created for military or advertising purposes, then probably mainstream serious game genres will be seen more often.

It can also be seen that entertainment RTS AR games, especially the tower defense sub-genre is rather popular. This sub-genre involves defending certain stationary assets by building more assets or shooting the enemies using screen controls. The reason why this is so popular might be that most AR games are moving towards marker-based technology, and the camera do not need to move much in tower defense games.

It also seems like nobody is making RPG AR games. The obvious reason might be that RPG games involve a lot of content, and it is not feasible to place thousands of markers in the real world. This is hence a shortcoming of marker-based AR games.

Insights

From the analysis described, several insights can be drawn for both research and commercial development.

In terms of research, it appears that the AR requirement of current software is very much built around the hardware that is provided. In current games, that means mostly detecting and tracking the markers. This, as mentioned earlier, has a scalability shortcoming. Other than that, designing and printing of markers is also a hassle. Hence a promising direction for research is to look into generic marker-less detection and tracking of generic objects. A good starting point would be to look at mature computer vision domains like face and gesture detection and recognition. There are marker-less AR research [8] that are on-going, just that nobody has applied it successfully in a working game.

Moreover, it is also observed that the serious AR games are mainly applied in the education and healthcare industries. Conventional serious games have been successful in military and advertising industries and hence AR versions of those games would prospectively be more engaging and fun as some of the researchers have shown.

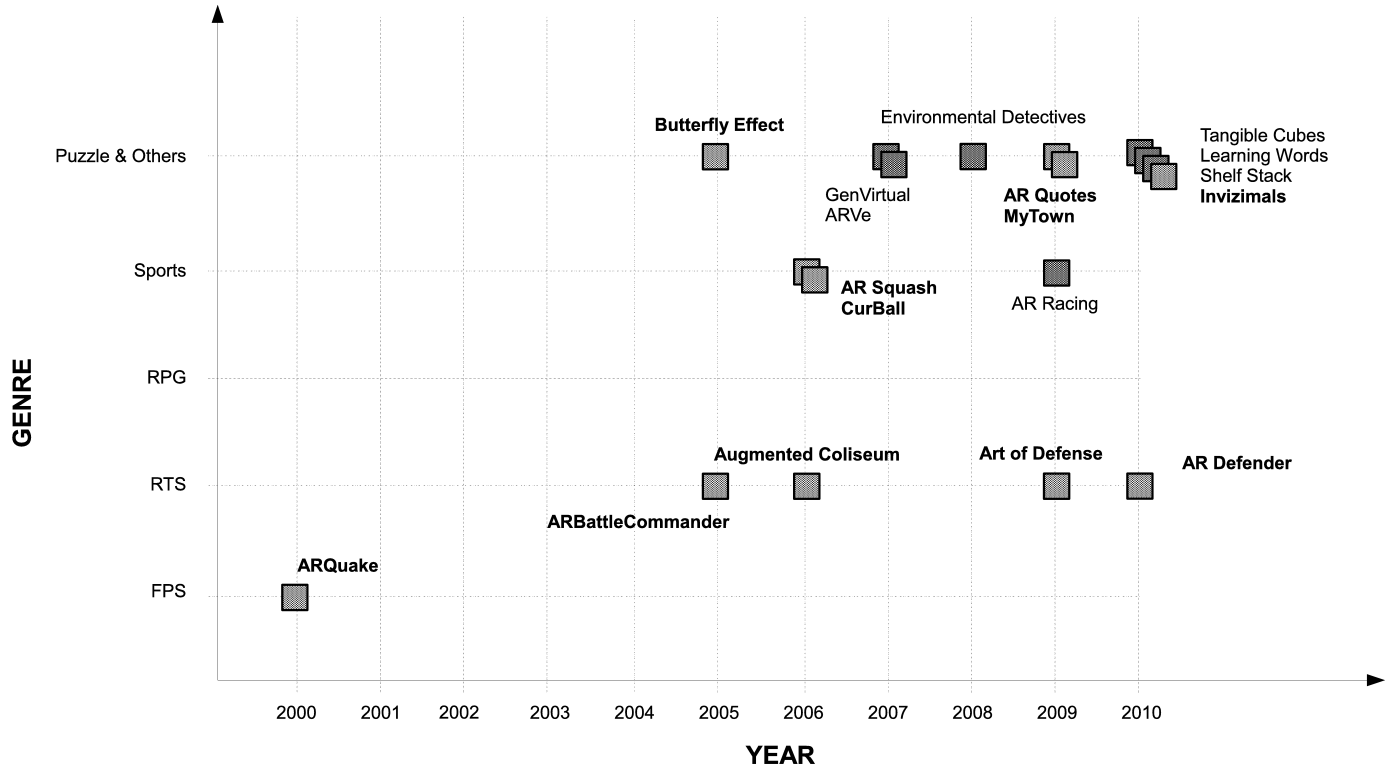


Figure 2: A graph of game genre versus year of release/publication. Blue color denotes entertainment games while red color denotes serious games. It can be seen that in general, AR games are largely non-mainstream genres. The only mainstream genre that is popular is that of tower defense RTS games.

This are also areas whereby research work can venture into.

In terms of commercial development, many unchartered domains exist that are yet to be commercially exploited. In commercial products, the first-mover's advantage is important to a product's success. Firstly, AR is not incorporated into any of the mainstream genres yet. Imagine for example Starcraft [7] being played by physically placing game cards on a map. Secondly, it seems like there are no serious AR games on the market currently. Most of the serious games created by the research community are technological demos at best, and it is up to the commercial developers to make these demos into professional solutions such that the society can benefit from them. Moreover, the advent of marker-based technology has made AR so much more accessible to consumers than it was previously, therefore this era would be a good time to make commercial AR games.

CONCLUSION AND FUTURE WORK

This paper has presented an initial effort at analyzing current AR games across various technologies used as well as game genres. The trend graphs show that modern AR games tend to make use of cheap and highly accessible equipment found on popular mobile consumer phones and laptops. They also show many avenues which are lacking in both research and commercial development. Hence researchers and game companies can make use of the review in this paper to better focus their energies in terms of AR games.

Although this paper tries to list most of the current state of the art AR games that have been created, it does not claim to have considered each and every single AR game created so far. This paper can be seen as simply a starting point at trying to consolidating an AR games database. Hence a future work is to create an automated database that people an submit a technological description of their AR games. In doing this, the trend graphs and certain aspects of analysis can also be automatically generated. Moreover, the current spectrum of analysis is currently only technology and game genre, hence more spectrums should be added in the future.

In all, this paper serves as an ignition for a continuous and iterative process of consolidating and analysing AR games in both research and industry, such that researchers and practitioners can advance better with a holistic view of the domain at all times.

REFERENCES

[1] Nate Anderson. Video gaming to be twice as big as music by 2011 (statistics from pricewaterhousecoopers), 2009. http://arstechnica.com/

gaming/news/2007/08/gaming-to-surge-50-|
percent-in-four-years-possibly.ars.

[2] Ronald Azuma. A survey of augmented reality. *Presence*, 6:355–385, 1995.

[3] Booyah. Mytown, 2010. http://www.booyah.com/.

[4] J.W. Burke, M.D.J. McNeill, D.K. Charles, P.J. Morrow, J.H. Crosbie, and S.M. McDonough. Augmented reality games for upper-limb stroke rehabilitation. pages 75 –78, mar. 2010.

[5] Ben Close, John Donoghue, John Squires, Phillip De Bondi, Michael Morris, Wayne Piekarski, Bruce Thomas, Bruce Thomas, and Unisa Edu Au. Arquake: An outdoor/indoor augmented reality first person application. In *In 4th Int'l Symposium on Wearable Computers*, pages 139–146, 2000.

[6] Ana Grasielle Dionisio Correa, Gilda Aparecida de Assis, Marilena do Nascimento, Irene Ficheman, and Roseli de Deus Lopes. Genvirtual: An augmented reality musical game for cognitive and motor rehabilitation. pages 1 –6, sep. 2007.

[7] Blizzard Entertainment. Starcraft, 2010. http://us.blizzard.com/en-us/games/sc/.

[8] V. Ferrari, T. Tuytelaars, and L. Van Gool. Markerless augmented reality with a real-time affine region tracker. In *PROC. INTL SYMPOSIUM ON AUGMENTED REALITY*, pages 87–96, 2001.

[9] Duy-Nguyen Ta Huynh, Karthik Raveendran, Yan Xu, Kimberly Spreen, and Blair MacIntyre. Art of defense: a collaborative handheld augmented reality board game. In *Sandbox '09: Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games*, pages 135–142, New York, NY, USA, 2009. ACM.

[10] id Sofware. Quake, 2010. http://www.idsoftware.com/games/quake/quake/.

[11] Ori Inbar. The first fun augmented reality game on the iphone app store was just submitted, 2010. http://gamesalfresco.com/2010/09/23/the-first-fun-augmented-reality-game-on|-the-iphone-app-store-was-just-submitted/.

[12] Amazon.com Inc. Bestseller in simulation, 2010. http://www.amazon.co.uk/gp/bestsellers/videogames/676515011/ref=pd_ts_vg_h__nav.

[13] Apple Inc. iphone, 2010. http://www.apple.com/iphone/.

[14] int13. Ar defender, 2010. http://www.ardefender.com/.

[15] Carmen M. Juan, Giacomo Toffetti, Francisco Abad, and Juan Cano. Tangible cubes used as the user interface in an augmented reality game for edutainment. pages 599 –603, jul. 2010.

[16] C.M. Juan, E. Llop, F. Abad, and J. Lluch. Learning words using augmented reality. pages 422 –426, jul. 2010.

[17] D. Kern, M. Stringer, G. Fitzpatrick, and A. Schmidt. Curball–a prototype tangible game for inter-generational play. pages 412 –418, jun. 2006.

[18] C. Kirner, E.R. Zorzal, and T.G. Kirner. Case studies on the development of games using augmented reality. volume 2, pages 1636 –1641, oct. 2006.

[19] Eric Klopfer and Kurt Squire. Environmental detectivesthe development of an augmented reality platform for environmental simulations. *Educational Technology Research and Development*, 56:203–228, 2008. 10.1007/s11423-007-9037-6.

[20] M. Kojima, M. Sugimoto, A. Nakamura, M. Tomita, H. Nii, and M. Inami. Augmented coliseum: an augmented game environment with small vehicles. page 6 pp., jan. 2006.

[21] Seok-Han Lee, Yong-In Yoon, Jong-Ho Choi, Chil-Woo Lee, Jin-Tae Kim, and Jong-Soo Choi. Ar squash game. pages 579 –584, sep. 2006.

[22] Fotis Liarokapis, Louis Macan, Gary Malone, Genaro Rebolledo-Mendez, and Sara de Freitas. A pervasive augmented reality serious game. *Games and Virtual Worlds for Serious Applications, Conference in*, 0:148–155, 2009.

[23] Sony Computer Entertainment America LLC. Playstation portable, 2010. http://us.playstation.com/.

[24] Sony Computer Entertainment Europe LLC. Invizimals, 2009. http://uk.playstation.com/psp/games/detail/item156403/Invizimals%E2%84%A2/.

[25] Pranav Mistry, Pattie Maes, and Liyan Chang. Wuw - wear ur world: a wearable gestural interface. In *CHI EA '09: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, pages 4111–4116. ACM, 2009.

[26] M. Norton and B. MacIntyre. Butterfly effect: an augmented reality puzzle game. pages 212 – 213, oct. 2005.

[27] Nintendo of America Inc. Wii remote controller, 2010. http://www.nintendo.com/wii/console/controllers.

[28] Keith Phillips and Wayne Piekarski. Possession techniques for interaction in real-time strategy augmented reality games. In *ACE '05: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, page 2, New York, NY, USA, 2005. ACM.

[29] Wayne Piekarski and Bruce Thomas. Arquake: the outdoor augmented reality gaming system. *Commun. ACM*, 45(1):36–38, 2002.

[30] E. Richard, V. Billaudeau, P. Richard, and G. Gaudin. Augmented reality for rehabilitation of cognitive disabled children: A preliminary study. pages 102 –108, sep. 2007.

[31] Brian Schwab. *AI Game Engine Programming*, chapter First-Person Shooters/Third-Person Shooters (FTPS), pages 123–142. Charles River Media, second edition, 2009.

[32] Brian Schwab. *AI Game Engine Programming*, chapter Real-Time Strategy (RTS) Games, pages 105–122. Charles River Media, second edition, 2009.

[33] Brian Schwab. *AI Game Engine Programming*, chapter Role Playing Games (RPGs), pages 69–92. Charles River Media, second edition, 2009.

[34] Brian Schwab. *AI Game Engine Programming*, chapter Sports Games, pages 171–190. Charles River Media, second edition, 2009.

[35] Robert Stone. Serious games: virtual reality's second coming? *Virtual Real.*, 13(1):1–2, 2009.

[36] Michael Zyda. From visual simulation to virtual reality to games. *Computer*, 38(9):25–32, 2005.

THE 6-11 FRAMEWORK: A NEW METHODOLOGY FOR GAME ANALYSIS AND DESIGN

Roberto Dillon
DigiPen Institute of Technology, Singapore
10 Central Exchange Green #01-01
Singapore 138649
Email: rdillon@digipen.edu

KEYWORDS
Game Development Methodology, Game Design and Research Methods, Gameplay Experience Evaluation.

ABSTRACT

This paper describes the “6-11 Framework”, a new game analysis and design methodology defined to help game designers in the process of crafting emotionally engaging and, ultimately, “fun” experiences. The framework is contextualized in an existing and well known model, the MDA, and then explained through relevant examples and case studies.

INTRODUCTION

Being able to quickly capture the essence of a game and understand what makes it "fun" is a required skill for any game designer. Unfortunately, due to the subjective nature of fun itself, too many analyses proposed so far tend to be quite subjective as well or to be applicable only in specific cases. Among the different approaches and models that have been proposed to explain why games can be so immersive and engaging, the MDA model (Hunicke et al. 2004) is the most well known and it is widely used across both the game industry and academia to analyze and formalize game ideas at different levels of abstraction. The MDA is based on the concepts of Mechanics, Dynamics and Aesthetics defined as follows:

- Mechanics: the game rules, i.e. basic, atomic actions that players can do to play the game.
- Dynamics: the gameplay, i.e. the complex actions that unfold as a result of applying the mechanics.
- Aesthetics: the emotional response evoked in the player, ultimately leading to a “fun” experience.

“Aesthetics” are the most challenging aspect to analyze, as they can be extremely variable and personal. The MDA model faces this issue by proposing the “8 Kinds of Fun”, a taxonomy introduced to explain different types of “fun” by relating them to experiences like problem solving (”Fun as Challenge”) or role playing (“Fun as Fantasy”).

This classification, while insightful and fascinating, provides only a very high level description of what is happening inside the players’ mind at an emotional level. In the end, it may not be very straightforward to relate a particular “kind of fun” to a specific in-game dynamic, especially for beginning game designers and students.

The "6-11 Framework" (Dillon 2010) tries to address these issues by providing a new taxonomy for game aesthetics which should be easy to relate to actual game dynamics. This process should result in a clearer and easier to understand picture of why a game is “fun” and how players’ emotional experience develops throughout the game.

THE 6-11 FRAMEWORK

The 6-11 Framework proposes that games can be so engaging at a subconscious level because they successfully rely on a subset of basic emotions and instincts which are common and deeply rooted into all of us.
Specifically, the framework focuses on six emotions and eleven instincts that are recurrent in psychology and widely analyzed in a number of well known treatises (Izard 1977; Plutchik 1980; Weiner and Graham 1984; Ekman 1999).

In particular, the six emotions are:

- Fear: one of the most common emotions in games nowadays. Thanks to the newest technologies, it is now possible to represent realistic environments and situations where fear can easily be triggered: think of all the recent survival horror games or dungeon explorations in RPG games for plenty of examples.

- Anger: A powerful emotion that is often used as a motivational factor to play again or to advance in the story to correct any wrongs that some bad guy did.

- Joy / Happiness: Arguably, one of the most relevant emotions for having a fun gaming experience. Usually this is a consequence of the player succeeding in some task and being rewarded by means of power ups, story advancements and so on.

- Pride: rewarding players and making them feel good for their achievements is an important motivational factor for pushing them to improve further and advance in the game to face even more difficult challenges.

- Sadness: Despite being an emotion that doesn't seem to match with the concept of "fun", game designers have always been attracted by it as a way to reach new artistic heights and touch more complex and mature themes.

- Excitement: most games worth playing should achieve this and it should happen naturally as a consequence of successfully triggering other emotions and/or instincts.

While the eleven core instincts taken into considerations are:

- Survival (Fight or Flight): the most fundamental and primordial of all instincts, triggered when we, like any other living being, are faced with a life threat. According to the situation, our brain will instantly decide whether we should face the threat and fight for our life or try to avoid it by finding a possible way of escaping. This is widely used in many modern videogames.

- Self Identification: people tend to admire successful individuals or smart fictional characters and naturally start to imagine of being like their models.

- Collecting: a very strong instinct that can link to a variety of different emotions and it has always been widely used in games.

- Greed: often we are prone to go beyond a simple "collection" and start amass much more than actually needed just for the sake of it. Whether we are talking about real valuable items or just goods and resources we need to build our virtual empire in a strategy game, a greedy instinct is likely to surface very early in our gaming habits.

- Protection / Care / Nurture: arguably the "best" instinct of all: the one that pushes every parent to love their children and every person to feel the impulse for caring and helping those in need.

- Aggressiveness: the other side of the coin, usually leading to violence when coupled with *greed* or *anger*. It is exploited in countless of games.

- Revenge: another powerful instinct that can act as a motivational force and is often used in games to advance the storyline or justify why we need to annihilate some bad guy.

- Competition: deeply linked with the social aspects of our psyche and one of most important instinct in relation to gaming, e.g. leaderboards. Without it, games would lose much of their appeal.

- Communication: the need for expressing ideas, thoughts, or just gossip, was one of the most influential for human evolution and it can be used to great effect in games too, while seeking information by talking to a non-playing character (NPC) or while sharing experiences with other players in chatrooms and forums.

- Exploration / Curiosity: all human discoveries, whether of a scientific or geographical nature, have been made thanks to these instincts that always pushed us towards the unknown.

- Color Appreciation: scenes and environments full of vibrant colors naturally attract us, including the more and more detailed and colorful graphics we see in modern games.

The main idea behind the 6-11 Framework is that these emotions and instincts interact with each other to build a network or sequence that should end with "Joy" and/or "Excitement", so as to provide players with a meaningful and fun experience. This network can then be related to game dynamics by realizing that, when different emotions are naturally aroused in the player by the game, these will trigger different instincts. These instincts, in turn, will force the player to act in the game, ultimately showing how the whole aesthetics can be linked to actual gameplay and game mechanics.

EXAMPLES

To gain a better understanding about the possible interactions between emotions and instincts we can describe by using this model, let us imagine we are designing a hypothetic action/adventure first person game where we start by immersing the player in a beautiful and colorful environment.
The luscious environment will naturally resonate with our "color appreciation" instinct, giving the player a sense of well-being and an early feeling of satisfaction/joy. This, together with the "self identification" instinct we stimulated through a proper first person perspective and, possibly, a compelling background storyline, will also help in triggering the player's "curiosity" which, in turn, will drive him to explore the surroundings. Once this happens, a "fearful" emotional state can be easily induced by a sudden encounter with a hostile creature, triggering the "survival" instincts with consequent "excitement" for the ensuing battle and confrontation.

The whole gaming experience, and the corresponding sequence of emotions and instincts, can be represented through a diagram, shown in Figure 1, where we see how specific emotions lead to instincts that, in turn, push the player to act into the game by means of the game dynamics, which are then made possible through particular mechanics. For example, here, our curiosity will make us explore the environment, walking around and eventually solving some environmental puzzle by opening/unlocking hidden passages, and then, once the dangerous encounter has been set, it will be our survival instinct to naturally drive our actions, for example by fighting the monster or by trying to escape to a safe area. Either way, the resulting experience and adrenaline rush will definitely bring excitement and, ultimately, deliver a fun experience to our brave adventurer.

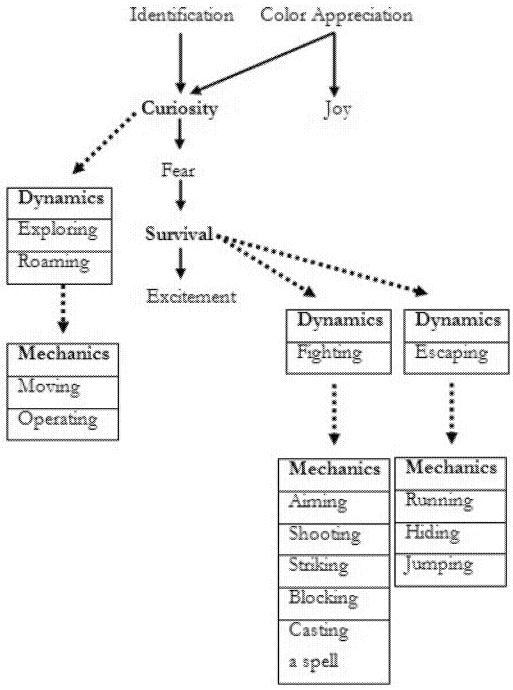


Figure 1: a diagram representing an analysis for a hypothetical action adventure game. Curiosity and Survival are the main instincts responsible for identifying the game aesthetics and are the main emotional factors driving the gameplay.

We can now try to analyze a commercial game, like the award winning mobile game "Angry Birds" (Rovio 2009) to see the framework at work on an actual, well known and best selling game.

In angry birds the player is tasked with the duty of driving a family of birds in their revenge against a group of pigs that stole their eggs. A well crafted cut scene starts the game by providing a very simple yet engaging background story to make the player sympathize and identify himself with the birds. This also fuels anger towards the pigs, motivating a revengeful feeling that will actually drive the whole playing experience through a carefully crafted dynamic of destroying the pigs' fortifications thanks to each bird's unique abilities (which will identify the individual game mechanics).

But there is more to analyze to fully understand what makes this game so compelling and engaging.

Besides very colorful graphics, which will help in increasing the player's curiosity to see the next scene and progress in the simple storyline, the possibility of recording our scores, on local and worldwide leader-boards alike, adds a very strong competitive element that will make players proud for their achievements and results. This pride will also be greatly enhanced by the possibility of collecting different bonus items, like golden eggs and stars, which will rise the overall score and will keep players interested for a long time.

By using the 6-11 framework, all this can be easily and concisely summarized in a diagram like the one in Figure 2.

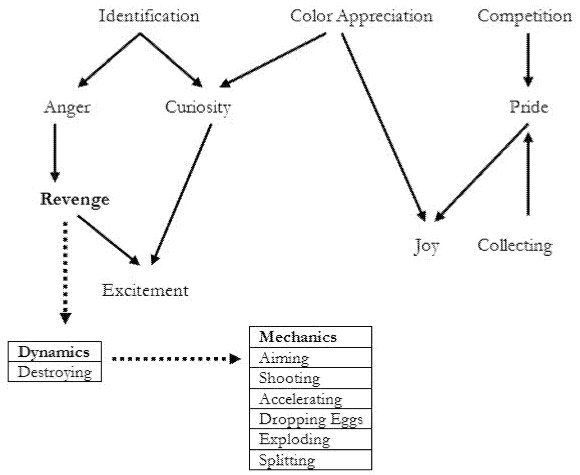


Figure 2: a diagram describing Rovio's game "Angry Birds". Several different instincts work together to deliver a very fun experience that is basically driven by the revengeful instinct to punish the pigs for stealing the birds' eggs. It's also important to note how the competitive and collecting instincts work together to make players proud for their achievements. This is fundamental to keep them engaged even once the action is over and motivate them to play again and improve their results.

CONCLUSIONS

The "6-11 Framework", especially when used in conjunction with the MDA model, seems like a very reliable approach for game designers to analyze games from many different genres. By explicating "aesthetics" in terms of familiar emotions and instincts, it can be used also as a design tool for organizing a game's inner working into a meaningful high level structure which will be more likely to subconsciously resonate with prospective players and successfully capture their total attention.

The proposed framework is also successfully used to teach game analysis in undergraduate level courses at DigiPen Singapore, providing students with a robust tool to break down any game into different layers of abstraction and then gain a proper insight not only on how a specific game works but also on the reasons why it is appealing to players and ultimately able to deliver a fun experience.

The use in class also provides plenty of opportunities for testing and validating the model in practice. Indeed, students' analyses shows that, while there can be slight variations from student to student, people tend to perceive a specific game experience in the same way. For example, when faced with a game like Battlezone (Atari 1980), most students easily identify a structure like the one represented in Figure 3, where the player gets immersed in a first person 3D environment and then needs to fight and escape from the enemy's line of fire.

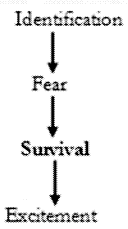


Figure 3: an analysis of the game aesthetics at play in Battlezone according to the 6-11 Framework

Similarly, when analyzing another classic arcade game such as Pac-Man (Namco 1980), all analysis identify how the fear to be "eaten" by the ghosts, together with the defenseless nature of our character, stirs our revengeful instincts as described in Figure 4.

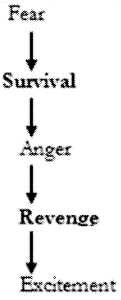


Figure 4: when analyzing Pac-Man, students identify a sequence where Fear and the following escaping action lead to a strong revengeful feeling which is then satisfied once the player is finally able to chase back the ghosts later in the game.

While more rigorous testing is certainly needed to validate the proposed model, these first results, showing common analyses and positive feedback from the students, seem to confirm that the set of emotions and instincts included are actually meaningful while also being general enough for describing games belonging to a wide variety of genres and styles.

REFERENCES

Dillon R. 2010: "On the Way to Fun: an Emotion Based Approach to Successful Game Design", AKPeters

Ekman P. 1999: "Basic Emotions", In T. Dalgleish and M. Power (Eds.). *"Handbook of Cognition and Emotion"*. Sussex, U.K.: John Wiley & Sons, Ltd.

Hunicke R., LeBanc M. and Zubek R. 2004: "MDA: a Formal Approach to Game Design and Research" In *Proc. 19th National Conference on Artificial Intelligence*, San Jose', CA

Izard C.E. 1977: "Human Emotions", Plenum Press, New York

Plutchik R. 1980: "A general psychoevolutionary theory of emotion", In *"Emotion: Theory, research, and experience: Vol. 1. Theories of emotion"*, Academic Press, New York

Weiner B., Graham S. 1984: "An attributional approach to emotional development", in *"Emotion, Cognition and Behavior",* Cambridge University Press, New York

WEB REFERENCES

Atari 1980: "Battlezone"
http://www.atari.com/play/atari/battlezone

Namco 1980: "Pac-Man"
http://en.wikipedia.org/wiki/Pac-Man

Rovio 2009: "Angry Birds",
http://www.rovio.com/index.php?page=angry-birds

BIOGRAPHY

ROBERTO DILLON was born in Genoa, Italy and holds a Master and a Ph.D. degree in Electrical and Computer Engineering from the University of Genoa. After having worked both in the software/multimedia development industry and in prestigious academic institutions across Europe and Asia, such as KTH in Sweden and NTU in Singapore, he joined the Singapore campus of the DigiPen Institute of Technology where he is currently an Assistant Professor lecturing a variety of game design subjects including "Game Mechanics" and "Game History". Prof. Dillon has led high profile research projects on innovative game mechanics and designed serious, educational and experimental games that were showcased internationally on newspapers like "USA Today" and at events like "Sense of Wonder Night" at the Tokyo Game Show.

WORK WITH MII: DISCIPLINING THE BODY IN THE WII FIT PROGRAM

Maria Emilynda Jeddahlyn Pia V. Benosa
University of the Philippines, Diliman, Quezon City, Philippines/
American University, Washington, D.C.
mvbenosa@up.edu.ph

KEYWORDS

Wii Fit, Nintendo DS, Neo-immersion, Game Impact

ABSTRACT

This paper is an analysis of the Wii Fit Plus experience and Wii Fit Plus player interview videos featured in the official Wii website, which presented the Wii Fit as a highly normative game that disciplines the body in subtle ways via immersion; if not punishing the bad obese body which exceeds its measuring limits, altogether. Using concepts from Whitson's study, this research reevaluates the Wii Fit's inclusion in the category of neo-immersive games, within which it has been lauded for aiding in a new generation of video game consoles presenting solutions to a sedentary lifestyle, and providing a space for mobility to gamers.

INTRODUCTION

The Wii Fit program has been regarded as a most welcome addition to the stock of neo-immersive videogames when it first came out in the market, for '[enabling] video gamers to employ active body movements [in] interaction mode' and for having a chance at 'becoming a global, affordable, mass-market interactive home fitness system' (Pasch 2008, Schiesel 2008).

Indeed, the success of the Wii Fit program, Nintendo Inc.'s current top-selling software, was not built solely on the revolutionizing technology of its console and the *Wiimote*, nor its dynamic set of exercises, but instead because '[it has been developed as an easy to use, inexpensive diversion for families]…rather than create a new generation of games that would titillate hard-core players', ushering in a slew of promotional materials like TV advertisements that showcased a 'pace that's friendly to girls and women and old people, as well as a slice of the 18–34-year-old male demographic' (Schiesel 2008, Jones 2008). In addition, whereas in traditional immersive games like World of Warcraft and Super Mario where users assume other pre-set selves and their characters' goals and motivations, neo-immersive games, like most Wii games, 'achieve an increased awareness of the physical self by foregrounding the importance of the game interface and controllers…allowing the story to recede in the background'—giving the appeal that the game is less than a game, but more like a real program (Whitson et al. 2008).

Unlike previous television commercials for the Wii Fit, however, where actors were silent, the Wii Fit Plus experience and Wii Fit Plus player interview videos featured in the official Wii website (www.wii.com), and which were analyzed for most of this research, now feature an array of characters who give testimonials on why exactly the Wii Fit program works for them. An analysis of these videos as well as the game's interface, guided by concepts from Whitson's study, has allowed for a redefinition of the very concept of neo-immersion in such a game as the Wii Fit.

AVATARS: HEY, THAT'S MII!

In the Wii platform, with the added sense of control that the Wiimote provides, avatars called Miis can be considered as virtually embodied as one can get in any game, as they are assumed to represent and approximate corporeal bodies. Even in the Wii Fit experience videos, one can see the striking resemblance of the interviewees to their Miis.

With avatars, the level of immersion of characters in the virtual world becomes closer to the feeling that "I am there" as opposed to "my character is in the game" (Castronova 2005). According to Peng, because there are usually set goals that players must achieve in games, 'these goals are also the goals of the game characters in the game environment' (2008). These statements could not ring any truer than in the case of the Wii Fit program. At the beginning of the game, when a Mii is first imported from the Mii channel to play in the Wii Fit program, the user is asked to perform a body test that will measure statistics on which the workout scheme for him will be based. As one weighs him or herself on the balance board, one can observe how the Mii either blows up or shrinks in size, depending on the user's body mass index (BMI). When a user exceeds the overweight limit and moves on to the realm of the obese, the Mii is shown as if in a desolate mood, and his virtual body is enlarged, even when a presumed height and width have been entered by the user during the creation of the Mii. What follows is a process where the console evaluates the statistics it has received and comes up with a usually amiss and exaggerated Wii Fit age, which it presents with accompanying drum roll and spotlights.

At this point, only when a user's Wii Fit age is similar to his real age that the Mii will be shown as happy and jumping in joy. Even a 2-year difference will show a Mii as utterly embarrassed and shaking his head. This scenario is similar to Yael Sherman's application of Foucault's *panopticon* where users are already outside their embodied

virtual bodies, and are directed in terms of how they should feel about the status being assigned to them by the equipment (Sherman 2008). Not only are users becoming docile bodies disciplined by the Wii Fit program, the users themselves move their gaze onto their new subjects: the Miis, which by virtue of transference of the characteristics of a corporeal body, is now in itself becoming a docile body that needs to be disciplined as well.

With this mode of instruction on how a user should feel about the results of a game or exercise, the Wii Fit becomes a normative tool that turns the users' motivations toward the idea of a "fit", non-obese body—the very normal, white, medium-built body that is the beginning template for all new creations of Miis. This begins a process of identification, wherein the user desires 'ability to increase the power of their avatars', or in this case, improve their Miis, as a representation of the improvement of their own bodies in the real world (Castronova 2005).

TOWARD A REDEFINITION OF IMMERSION

It is its neo-immersive gameplay that makes the Wii Fit a pioneer in the field of 'movement-based consoles [that] reach exertion levels that increase physical health and appear promising for reducing obesity, which is at least partly the result of a sedentary lifestyle' (Pasch 2008). By giving its users the promise of health and fitness as long as one sticks to one's two-week goals and follows the balance board's virtual advise to "Keep hydrated and eat fruits!" users and new buyers alike are drawn to this device that could help their previously inactive bodies.

Indeed, it is highly ironic that a video game console such as the Wii Fit, should present itself as a solution to the sedentary lifestyle of which one major cause is the very technology sold by computer games, especially as gaming technologies become more and more sophisticated—both in isolating gamers from the real world (therefore immersing them in the game), and also in incorporating their actions in the real world to the screen.

Whitson's definitions of immersive and neo-immersive gameplay were fundamental to the findings of this research. According to Whitson's study, neo-immersive games, the category under which the Wii Fit program supposedly belongs to, 'start with the body, building upon the embodied knowledges and sets of experiences that users already possess, and that connect them to social networks and contexts. . .[Neo-immersion] acknowledges the body and its various existing contexts as an integral part of game design. This inside-out approach adapts technology to the needs and lifestyles of users, rather than demanding their full attention and immobilization' (Whitson et al. 2008).

On the other hand, they define immersion as 'the encapsulating of the physical body in technology and representations. In this model of immersion, the user's body is treated as an object, hooked up and forced to adapt to a technology that is socially alienating and profoundly solipsistic, and that rarely provides opportunities for engaging social and bodily interactions. Loss of self-awareness, social awareness and game awareness are required to keep the user in a state of flow and within a seamless magic circle of representation' (Whitson et al. 2008).

The Wii Fit program, however, stands at the middle ground of these categories, and to a certain degree is innovating on the idea of 'immersion' to create a new form of immersion, or simply, a new neo-immersion.

To begin, the Wii Fit definitely encloses the physical body in technology and representations. Whitson notes that 'the Wiimote [has become] a kind of enabling or prosthetic technology that enhances the physical body, rather than transcending it through out-of-body experiences' (Whitson et al. 2008). Not only are Wii Fit routines abrupt and discontinuous—diminishing the semblance of working out with it with real, physical "workout"—for some supposedly strenuous Wii Fit routines like boxing or running, 'it is sufficient to make a small movement from the wrist' to succeed (Pasch 2008). In other words, the nature of the Wii Fit as a videogame itself, is not far-fetched from the selection of videogames that contribute to the current generation's sedentary lifestyle.

Also, there is the users' preoccupation with the cute and charming Miis. Because the emotional response elicited in their countenance instruct users how to feel, they go through a transcendence from the real world, to a passive state where there is a 'loss of self-awareness, social awareness and even game awareness' the very characteristics that identify immersive gameplay (Whitson et al. 2008). In fact, what can be more passive and lazy than the following statements from interviewees in the Wii Fit Plus experience videos: "I like exercising at home because I could be dressed however I want. I could just take my time. If I get a phone call I could stop for a minute, go and continue when I come back. … A new feature that I really like are the routines, they have it all preplanned for you so they'll just target whatever, and you don't have to think about it" (Appendix, Video 2).

The parody video from SarcasticGamer.Com sums this observation up and says in an inviting tone, "Find out today why people all over the world think they're getting exercise with that little white thing you stand on. It's simple. It's easy. And you don't have to do anything" (Appendix, Video 1).

While it is true that a simulator game like The Sims by Electronic Arts is traditionally considered the more immersive game, the Wii Fit program actually follows closely in its ranks, but is marching in with a subtle and different process of alienation, as well as a reinforcement of solipsism. Both games do not follow a narrative, the decision of how the game progresses is entirely up to the user and his utility of his Mii. Unlike The Sims, however, the synthetic world of the Wii Fit does not compare well to the real world, or even attempt to be mimetic. "On earth, we have trees and buildings but they are not always designed to please the eye" (Castronova 2005). Wii Fit eliminates things and circumstances that would otherwise prevent the Wii Fit's very users from working out in the real world—by incorporating a suite of all exercises, routines and the personal fancies a user thinks he or she needs, without enduring subjection to the gaze, as when he

or she is in a gym. This is especially true for the already subjected body, one that is outside the presumed normal age or weight group that are socially acceptable to work out—including the obese body, which the advertising materials of the Wii Fit so blatantly avoid. Thus, in the privacy of the Wii Fit program, such outcast, disabled, dismembered bodies, can find refuge—an escape with an unclear promise of what really will emerge. Castronova says, in the closing of his chapter on the Wii platform, 'Thus, synthetic worlds may find their significance, not in being radically different from the world of ordinary life, but in being so similar to it as to present a comparable option, one that in some cases may be the better of the two' (2005). The Wii Fit has ensured, through its promotional materials, that the Wii Fit experience is a private one—that it can be best enjoyed in the private space that is the home, in an inclusive activity that requires no interaction with persons other than family, or a select group of friends, in a safe space where no actual harm can be incurred, and in a privileged environment where the best utility of a gaming console can be attained by obtaining classy furniture, electronic appliances and high-end fitness apparel.

CONCLUSION

Even the nature of the Wii Fit as an endless game, like The Sims, is suspect. Castronova says of these long-run projections, 'It seems that, unlike gory shooter video games, skateboards and punk music, there is no obvious limit to this growth, no point at which one can say, "Even the truest fan will give this up and start living like the rest of us,"' (2005). The Wii Fit program, in other words, is a forever thing. A player's attachment to the program depends then on when the target weight is reached—if it is ever reached with the games which are at best, dependent on wrist and hip power—and how often the person's weight dwindles and moves back into the realms of the overweight and the obese. By keeping these bodies subject, and inside their homes—further isolating them from the outside world, one's engagement with the Wii Fit is part of a series of consumption patterns based on the pretense of self-improvement. As sure as the Wii Fit is a "forever thing", one can also be certain that there will be more upgrades for the software to come. One of the most recent developments in the Wii platform is that with the balance board, one can now opt for a more strenuous workout software tailored by *The Biggest Loser* star trainer Jillian Michaels herself.

Perhaps, after all this talk, the most feasible potential site (or threat, depending on which side one is on) for the Wii Fit program to become an ultimate example of a new neo-immersion, is in the Mii channel, the virtual location in the gaming console where Miis are allowed to interact with each other. Steven Jones says, "Miis are one of the key identifying features of the system. Once you make them they live in the console's memory but show up in many Wii games, either as playable characters or non-player characters (for example, as team-members or cheering spectators in a sports game). On the Mii Plaza they can mingle online (in a Mii Parade) with your own and other people's Miis" (Jones 2008).

Though there has been no mention of it in press materials, and it might still be a long way from where we are today, there exists the possibility that in the future, with the utilization of the internet, the Wii Fit will join the ranks of online games where users can interact with each other, a tradition of gaming that was only previously and traditionally exclusive to PC games. It is a far-fetched idea, but it is also an extreme one that when developed may defeat the very purposes for which the Wii Fit program exists and has become so successful for: the absence of a sense of appreciative community in which outcast bodies are able to examine each others' narratives, paving the way for them to improve their sense of selves in a non-virtual way: one that is not dictated by anyone, or a virtual identity like a talking balance board or yoga trainer, even one that does not necessarily succumb to the umbrella of the perceived concept of "fitness".

APPENDIX: VIDEOS

1. Nintendo, Inc. "Wii Fit Plus Experience 2." 26 October 2009. Online video clip. Nintendo Wii . Accessed on 13 February 2010. http://us.wii.com/viewer_wii fitplus.jsp?vid=2
2. Nintendo, Inc. "Wii Fit Plus Experience 3." 26 October 2009. Online video clip. Nintendo Wii Official Site. Accessed on 13 February 2010. http://us.wii.com/viewer_wii fitplus.jsp?vid=3

REFERENCES

Castronova, Edward. "Synthetic Worlds: The Culture and Business of Online Games." Chicago: University of Chicago Press, 2005. 51-52, 72, 78.

Jones, Steven. 2008. "The Meaning of Videogames: Gaming and Textual Strategies." NY: Routledge. 135, 145.

Sherman, Yael. 2008 "Fashioning Femininity: Clothing the Body and the Self in *What Not to Wear*." Ed. Gareth Palmer. Surrey: Ashgate Publishing Ltd. 49-63.

WEB REFERENCES

Pasch, Marco, et. Al. 2008. "Motivations, Strategies and Movement Patterns of Video Gamers Playing Nintendo Wii Boxing." *2008 Facial and Bodily Expressions for Control and Adaptations of Games.* Amsterdam: University of Twente UP. 29-35. 11 February 2010.

Peng, Wei. 2008. "The Mediational Role of Identification in the Relationship between Experience Mode and Self-Efficacy: Enactive Role-Playing versus Passive Observation." *CyberPsychology & Behavior* 11.6: 649-652. 11 February 2010.

Schiesel, Seth. 2008. OK Avatar, Work with Me. *The New York Times*. Retrieved from http://www.nytimes.com/2008/05/15/fashion/15fitness.html. 11 February 2010.

Whitson, Jennifer, et. Al. 2008. "Neo-Immersion: Awareness and Engagement in Gameplay."*2008 Future Play: Research,Play,Share*. Association for Computing Machinery. New York: ACM. 223. 11 February 2010.

BIOGRAPHY

Maria E. J. Pia V. Benosa is finishing her BA in Creative Writing from the University of the Philippines. This paper was written while she was under a U.S. Department of State grant at American University, Washington, D.C. in 2010.

GAME AI

A MODEL FOR VISITOR CIRCULATION SIMULATION IN SECOND LIFE

Kingkarn Sookhanaphibarn
Intelligent Computer Entertainment Laboratory
Dept. of Human and Computer Intelligence
Ritsumeikan University
1-1-1, Noji-higashi, Kusatsu, Shiga, Japan 5258577
Email: kingkarn@ice.ci.ritsumei.ac.jp

Ruck Thawonmas
Intelligent Computer Entertainment Laboratory
Dept. of Human and Computer Intelligence
Ritsumeikan University
1-1-1, Noji-higashi, Kusatsu, Shiga, Japan 5258577
Email: ruck@ci.ritsumei.ac.jp

Frank Rinaldo
Intelligent Computer Entertainment Laboratory
Dept. of Human and Computer Intelligence
Ritsumeikan University
1-1-1, Noji-higashi, Kusatsu, Shiga, Japan 5258577
Email: rinaldo@is.ritsumei.ac.jp

Nadia Magnenat Thalmann
Institute for Media Innovation
Nanyang Technological University
50 Nanyang Drive, Research Techno Plaza
XFrontiers Block, Level 03-0, Singapore 637553
Email: nadiathalmann@ntu.edu.sg

KEYWORDS
Nonplayer character (NPC); OCC model; Personality and emotion simulation; Visitor circulation; Crowd simulation; Second life (SL); Computer game; Virtual world

ABSTRACT

The realistic places in SL require not only immersive content but also a crowd such as virtual classmates in a school, customers in a shopping mall, visitors in a gallery, and so on. With a great number of successful online role-playing games, we introduce a model-based approach in consideration of personality, emotion, and mood for controlling NPC circulating in SL. Merits of this model are shown as follows: First, NPCs can express their emotion to an environment as human being. Second, NPCs can travel to place by place without the human control. Lastly, it increases a degree of realization of virtual places in SL because a number of NPCs makes them lively.

INTRODUCTION

Second Life (SL) is a virtual 3-D world bursting with opportunities in areas like collaborative learning, legal practice, corporate connections, and people.Virtual World, also known as Synthetic World, is a computer-simulated persistent environment similar to the Real Life (RL). Users in SL are called as residents which are represented by avatars. Like most of other virtual worlds, SL can support massively multiplayer to be online at the same time. SL provides a wide spectrum of online activities, including arts, science, sports, and education. Within it, the residents can explore, meet other residents, socialize, and participate in all kinds of activities. The created places such as virtual museums, galleries, exhibitions and other showrooms require the existence of either virtual crowds or virtual participants in order to making their presence felt.

Similar to Massively Multiplayer Online Role-Playing Games (MMORPGs), their key factor is a cast of NPC whose characters are assigned to act as enemies, partners, or supporters of players. These NPCs make human players enjoyable and excited with a coherent storyline. These characters exist in a persistent virtual world as the same time as human players take on roles such as warriors, magicians and thieves and play and interact with NPCs and each others. These casts of the NPCs play an important role to make a game story sound.

In the real world, most of popular museums have their repelling atmosphere. In this paper, we aim at simulating visitor circulation in SL as similar to human-like behaviors while they are shopping in a department store, observing artworks in a gallery, and visiting an exhibition event. To achieve this goal, we develop a model integrating with the concepts of psychological models and the theories related to visitor behaviors.

Unlike computer games played in non-persistent worlds, SL and persistent game worlds present a game play over months rather than hours. This must be supported by NPCs. However, most of current NPCs trend to constrain them to a set of fixed behaviors unable to be evolved in time with their experience. Then, Merrick [Merrick, 2008] proposed a technique named *Motivated Reinforcement Learning (MRL) agent* offering an alternative to this type of character. MRL agents offer a motivation process to identify interesting events which are used to calculate a reward signal for a reinforcement learner. MRL agents are able to continually identify

new events on which to focus their attention and learn about.

We propose a model of visitor circulation simulation in virtual worlds in the same fashion of NPCs developed in MMORGs. Basically, the emotion is one of driven factors of human behaviors in any situations. For example, at art gallery, a visitor walking slightly fast and stopped at a few artworks in the first ten minutes, but stopped and observed all the rest artworks. In the psychology, these behaviors can be described with the OCC and OCEAN models. Our model integrates the revealing visit styles in real museums [Sparacino, 2002, Chittaro and Ieronutti, 2004, Sookhanaphibarn and Thawonmas, 2009] as the principal behaviors with the OCC model. This integral components make our model different from others.

Personality and Emotion Models

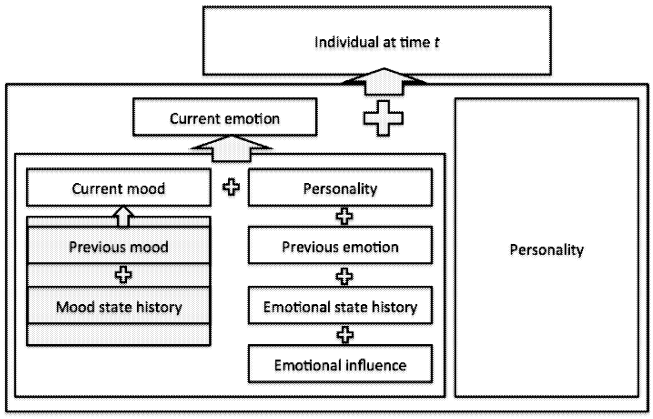


Figure 1: An abstract entity $I(p, e_t)$ of an individual at time t consisting of personality p and emotional state t

Figure 1 illustrates an abstract entity of an individual who will express his/her emotion based on the current mood and personality. This diagram is a graphical explanation of the personality, emotion and mood model as introduced in [Egges et al., 2004].

Personality

Basically, personality is used to set the threshold to generate emotional states and control the intensities. Information about the characters' personality can influence the probability of characters' behavior. Egges et al. [Egges et al., 2004] developed a small interaction system that simulates Julie's behavior. Given that Julie has an introvert personality, she will feel fear and distress and she will respond quite aggressively to a stranger. In the opposite, if Julie has an extravert personality, she will happy and she will accept a help from the stranger. The above situation illustrates how the personality designs the characters' behavior. As the above given situation, introversion is one of five factors referred to the OCEAN model as shown below:

- Openness influencing characters to have many interests but with less depth in each interest.
- Conscientiousness making characters to focus on less goals and exhibits self-discipline.
- Extraversion allowing characters to make more friends, comfortable with a higher number of social relationships.
- Agreeableness supporting character to accept the groups norms with ease and is termed an adapter.
- Neuroticism, associated with high levels of anxiety, stimulating characters to be reactive.

Emotion

The generation of emotions can be defined using the OCC with 22 emotion types [Bartneck, 2002]. The hierarchy contains three branches, namely emotions concerning consequences of events (e.g., joy or pity), actions of agents (e.g. pride or reproach), and aspects of objects (e.g. love or hate). The agent learns about the properties of different events, actions of agents, and aspects of objects through reinforcement learning and about the user through a probabilistic model that keeps track of their actions. The learning model significantly improved the plausibility of the affective behaviors expressed by the agents. Merrick and Maher [Merrick, 2008] simulated natural emotional expression through the modeling of the OCC model using neural networks.

Mood

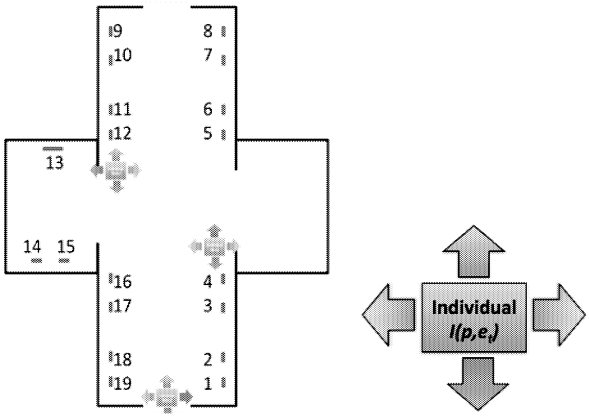
Mood represents the overall view of an individuals internal state. Other than functionality, an affective state is differentiated as an emotion or a mood based on three other criteria: temporal, expression and cause. Emotions are brief lasting for a matter of seconds or at most minutes. Moods last for longer, are not associated with a specific expression or cause. The main functional difference between emotions and moods is that, emotions modulate actions while moods modulate cognition. For instance, an individual in an irritable mood becomes angry more readily than usual and the resulting anger is more intense, decays more slowly and is more difficult to control.

VISITOR CIRCULATION SIMULATION

To simulate visitor circulation, we require the particulars as shown in Figure. 2. The simulation of visitor circulation is a process of decision making for a direction of an NPC going ahead. As shown in Figure 3, an individual will compute their emotion based on their

- Information of exhibited items in a virtual place consists of
  - Set of locations accessible to them.
  - Set of neighboring exhibited items.
  - Displaying properties of items such as shape, texture, color and size.
  - Description of items.
- Information of previous visitors' observation.
  - Time duration of visitor staying at each location (duration).
  - Sequence of items where each visitor followed.
- Current information of all visitors, i.e., NPCs and human players.
  - Degree of crowded for all items.
  - Visiting type of all visitors.

Figure 2: Information for visitor circulation simulation



(a) Three pivoting locations (b) An individual

Figure 3: Decision at pivoting locations

current state of and the history state of mood, and their personality as shown in Figure 1.

Given three pivot situations as shown in Figure 3 (a), for example, in the SL kimono gallery most human beings will make a decision of changing their direction whenever they found an exit and an entrance having two turn directions. As shown in Figure 3, all item number $1-19$ are kimono textiles exhibited in SL and its ordering is assumed as a predefined route. Figure 3 (b) represents a decision of what direction an individual $I(p, e_t)$ is going next.

First situation at gallery entrance, supposed that there are kimono items at both walls. Following the turn right bias as explained by Bitgood [Bitgood, 1998], the probability of turning right for some types of personality should be greater than the other directions.

Second, visitors with particular emotions such as getting tried or bored must consider to change their direction when facing an exit after item #4. Since there are none of items on the turning right, the dominant direction is still the going straight.

Lastly, for another of an exit-found situation near item #12, most visitors with a particular personality [Chittaro and Ieronutti, 2004] will exit to see the other items #13, #14, or #15.

CONCLUSIONS AND FUTURE WORK

We proposed a novel model of visitor circulation simulation and introduced a strategy of using NPCs to promote virtual museums and galleries in SL. For the future work, we will investigate the significance of potential factors such as dwell time, mouse movements, avatar movements, etc. The evaluation can be achieved by either explicitly interviewing a group of users or implicitly capturing their emotional feeling by the motion capture. Without loss of generality, we will conduct the experiment in the virtual Kimono gallery, owned by Ritsumeikan University, in SL.

REFERENCES

C. Bartneck. Integrating the OCC Model of Emotions in Embodied Characters. In *Proceedings of the Workshop on Virtual Conversational Characters: Applications, Methods, and Research Challenges, Melbourne.*, 2002.

S. Bitgood. An overview of the methodology of visitor studies. *Visitor Behav.*, pages 4–6, 1998.

L. Chittaro and L. Ieronutti. A visual tool for tracing users' behavior in virtual environments. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '04, pages 40–47, 2004.

A. Egges, S. Kshirsagar, and N. Magnenat-Thalmann. Generic personality and emotion simulation for conversational agents. *Computer Animation and Virtual Worlds*, 15(1):1–13, January 2004.

K. Merrick. Modeling motivation for adaptive nonplayer characters in dynamic computer game worlds. *Comput. Entertain.*, 5:5:1–5:32, March 2008.

K. Sookhanaphibarn and R. Thawonmas. A content management system for user-driven museums in second life. *Cyberworlds, International Conference on*, pages 185–189, 2009.

F. Sparacino. The museum wearable: real-time sensor-driven understanding of visitors' interests for personalized visually-augmented museum experiences. In *Proceedings of Museums and the Web (MW2002*, 2002.

IMPROVED PARETO OPTIMUM PASSING USING VARIED KICKING SPEED IN SOCCER GAMES

Nattawit Tanjapatkul and Vishnu Kotrajaras
Department of Computer Engineering
Faculty of Engineering, Chulalongkorn University
Payathai Road, Patumwan, Bangkok 10330, Thailand
E-mail: nattawitta@gmail.com, Vishnu@cp.eng.chula.ac.th, ajarntoe@gmail.com

KEYWORDS
Soccer Games, Pareto Optimality, Multi-Objective Optimization Problem, Artificial Intelligence.

ABSTRACT

Ball passing is very important in soccer games. A good pass to a teammate can create a scoring chance, which is critical for winning the game. This paper presents the improvement of ball passing for Artificial Intelligence (AI) in soccer games that uses pareto optimum passing. We have developed our own soccer simulation using Unity3D game engine. An approachfor finding suitable values for risk parameters on each position on the pitch is proposed. We also introduce the algorithm for calculating a suitable kicking force in order to reduce risk value of positions on the pitch, allowing previously too risky positions to be included in the passing decision algorithm. The results show that AI using the proposed algorithm has better scoring chances and wins more matches than the same AI that does not recalculate the kicking force. This proves our methodology can increase the performance of ball passing for AI in soccer games.

1. INTRODUCTION

AI in soccer games must behave rationally and must collaborate with its teammates to play soccer like a real soccer player. Decision making is vital when AI is controlling the ball. It must decide which action to perform such as passing, dribbling or shooting. Passing is one of the most crucial aspects of soccer games. AI must know how and where to pass the ball. Dribbling and shooting can also be considered as passing the ball to oneself and to the goal respectively. The advantage for generalising these actions as passing is to allow simplifications in the implementation. The code for dribbling and shooting can therefore use variations of passing. No specialized algorithm needs to be created for dribbling and shooting.

Previous research about this topic can mostly be found in RoboCup. Although designing AI for digital soccer games does not have as many restrictions as designing AI for RoboCup, the same concepts can be applied. One of the most effective approaches for the ball-passing problem in RoboCup used evaluation functions to evaluate risks and rewards when passing to each position on the pitch (Reis and Lau 2001; Stone and McAllester 2001; Kyrylov 2007; Yuan and Yingzi 2008). Kyrylov applied Pareto optimality to the ball-passing problem (Kyrylov 2007). The ball-passing problem was treated as a multi-objective optimization problem. The objectives of the problem were broken up into gains, risks and costs. The method for solving the multi-objective optimization problem was to find solutions which had at least one objective that was better than other solutions. These solutions were called Pareto optimum. The term 'optimum' did not mean that the final solution was the best. It meant that the final solution had the values of all the objective functions acceptable to the decision maker (Coello Coello, Lamont and Van Veldhuizen 2007). The soccer field was divided into many grid points which represented target positions that the Pareto optimum AI could pass the ball to. Each position on the soccer field had its own parameters of gains, risks and costs. Those values would be calculated and evaluated by the Pareto optimum AI in order to find the Pareto optimum position. With good weight balance between gains, risks and costs, the Pareto optimum AI was able to make effective decisions. However, the behavior was tested only on half the field. No full competition was evaluated. Despite such drawback, we believe the method to be suitable for developing AI in digital soccer games because of its scalability and robustness.

This paper presents an enhanced Pareto optimum passing algorithm based on Kyrylov's (2007). However, instead of assuming the kicking force to be constant, we incorporated varied kicking force when calculating risks for each position on the field. Our approach allowed some previously rejected positions to be considered for ball passing, leading to better plays and more goalscoring opportunities.

In Section 2, we briefly discusses our decision for developing a new testing environment. Section 3 presents our risk functions and our kicking force adjustment algorithm. Our experiments and results are shown in Section 4. Section 5 gives conclusion and future works.

2. SOCCER SIMULATION TESTBED

We have developed our own soccer simulation using Unity3D game engine (see Figure 1). Simulation environments became easier to control compared to using the RoboCup soccer server because we were not forced to deal with restrictions not necessary for commercial soccer games, such as players' limited field of vision, physics environment, and communications (Noda, Matsubara,

Hiraki and Frank 1998; Bustamante, Flores and Garrido 2007; Yuan and Yingzi 2008). Unity3D was chosen because of its ease of use and its support for many useful features such as built-in world builder, profiler, networking and scripting. The community was also very helpful and there were many documents and tutorials for beginners. Our soccer field has 6500 target positions in total for the evaluation of passing . Due to limited space allowed for this paper, we cannot further discuss the environment at this time. Readers who are interested in our testbed can download source code from http://www.cp.eng.chula.ac.th/~vishnu/

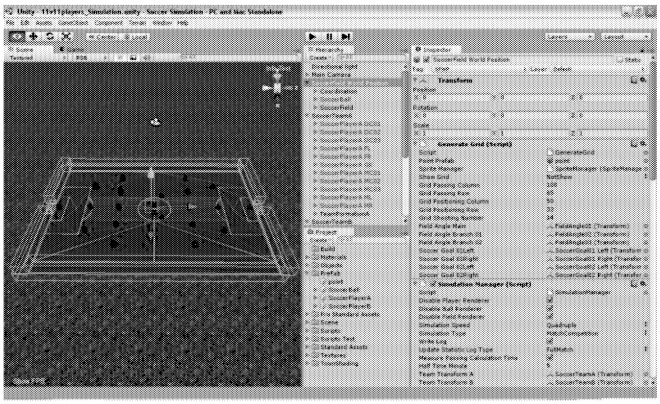


Figure 1: Soccer Simulation Developed in Unity3D

3. BALL PASSING DECISION

3.1 Risk Functions And Constraints

All risk functions were similar to Kyrylov's (2007). A risk function was used to determine whether the ball could be passed safely to a given position according to a given safety threshold. For each passing opportunity, all risk functions of each position on the field were calculated and if one or more of them did not pass their safety threshold, the position would be removed from future Pareto optimality calculation (Pareto optimality calculation used weighed risks, gains, and costs). Each risk is summarized below to get readers familiar with our approach.

3.1.1 An opponent may reach the intended passing position *(x,y)* before AI's teammate. Hence, the risk function $r_{\text{opponent\_reach}}(x,y)$ is the time difference between the arrival of the first possible teammate at position *(x,y)* and the arrival of the first possible opponent at position *(x,y)*.

3.1.2 The ball can be intercepted by an opponent while on its way to position *(x,y)*. Therefore, the risk function $r_{\text{intercepted}}(x,y)$ is the time difference between the intended arrival time of the ball at (x,y) and the earliest time it can be stolen by an opponent.

3.1.3 A teammate may arrive at position *(x,y)* some time after the ball moves away from the position. Consequently, the risk function $r_{\text{ball\_too\_fast}}(x,y)$ is the time difference between the arrival of the first possible teammate at position *(x,y)* and the arrival of the ball at position *(x,y)*.

3.1.4 Too many opponents maybe too close to position *(x,y)*. As a result, the risk function $r_{\text{many\_opponents}}(x,y)$ is the time difference between the arrival of the ball at position *(x,y)* and the arrival of the second possible opponent at position *(x,y)*.

3.1.5 If all teammates fail to intercept the ball, it may cross the field boundary. Therefore, the risk function $r_{\text{out\_of\_field}}(x,y)$ is the negative value of the time the ball takes to cross the field boundary after leaving position *(x,y)*.

We left out two risk functions from Kyrylov's work because they dealt with noise compensation and player stamina. These were not our concerns since we were trying to simulate an action game, not making a soccer robot simulation. This allowed us to fully control the environment and accurately evaluate our method without having to worry whether results were affected by those parameters.

The maximum acceptable value of each risk function was called a risk constraint. All risk constraints were adjustable in the GUI of our simulation editor. Risk constraints directly affected passing decisions of our AI. Any position failing one of its constraints would not be included when calculating the Pareto optimality position. A high value for risk constraints for $r_{\text{many\_opponents}}$ would scare our AI away from passing the ball up the field. The AI would consider the opponent's half of the field to be too dangerous because there were opponent players ready to intercept the ball. A low value for risk constraints would encourage the AI to make reckless passes that could easily be intercepted by opponent players. Ideally, all risk constraints should have negative values. However, such set-up did not work well when we carried out experiments because the AI kept exhibiting the same behavior as when using high risk constraints. In order to find appropriate values for the risk constraints, experiments had to be carried out. Our experiments for finding these values are discussed in Section 4.

3.2 Changing Ball Speed

The 'ball destination speed' refers to the speed of the ball when it reaches its destination. This speed depends on the kicking force used by the sender of the ball. In our research, we introduced an algorithm for varying the ball destination speed. The AI that used unfixed ball destination speed was then compared with the AI that still maintained constant ball destination speed.

3.2.1 Fixed Ball Destination Speed

The AI that used fixed ball destination speed was originally presented by Kyrylov (Kyrylov 2007). However, it was not clear how the value of ball destination speed was achieved. Therefore we had to come up with our own approach. Our assumption was that the travelling speed of the ball between the time it was kicked and the time it reached its destination should at least be faster than the speed of other players (Buckland 2005). This prevented players from overtaking

the ball from behind. Based on this assumption, the ball destination speed was set to the maximum speed of opponent players. With known destination speed value, suitable kicking force was calculated.

3.2.2 Unfixed Ball Destination Speed

This was our methodology for improving the AI that utilized fixed ball destination speed. When making a decision on passing, it was better to have more passing destinations because the AI was able to evaluate more potential alternatives in order to search for an optimum point. Passing the ball with fixed ball destination speed had its limitation. It prevented the AI from selecting some high risk passing destinations that had good reward or good tactical values. However, if the kicking force could be increased, the values of risk functions $r_{intercepted}$ and $r_{many\_opponents}$ of those destinations could be reduced, allowing them to be selected for passing. But increasing the kicking speed also gave greater risk values for functions $r_{ball\_too\_fast}$ and $r_{out\_of\_field}$. This might again lead to some passing destinations being too risky to consider. Consequently there was a limit to the power we were able to increase for each kick.

The implementation idea behind unfixed ball destination speed was to find the minimum and the maximum kicking force that did not cause every risk value to exceed its constraint. The pseudocode for this method is shown below.

Algorithm 1: Pseudocode of Calculating Kicking Force From Risk Constraints for Unfixed Ball Destination Speed

```
maxKick ← playerMaxKick
minKick ← playerMinKick

tempKick ← compute: kicking force that causes r_out_of_field to
           reach its constraint value
if tempKick < maxKick then
  maxKick ← tempKick
  if maxKick < minKick then
    return //neglect this passing destination
  end if
end if

tempKick ← compute: kicking force that causes
r_many_opponents
           to reach its constraint value
if tempKick > minKick then
  minKick ← tempKick
  if maxKick < minKick then
    return //neglect this passing destination
  end if
end if

tempKick ← compute: kick force that causes r_ball_too_fast to
           reach its constraint
if tempKick < maxKick then
  maxKick ← tempKick
  if maxKick < minKick then
    return //neglect this passing destination
  end if
end if

maxKick ← round value down
minKick ← round value up

increment ← 1
tempKick ← minKick
neglect ← true

while tempKick <= maxKick
  r_intercepted ← compute: r_intercepted using tempKick
  if r_intercepted <= constraint_of_ r_intercepted then
    if tempKick > minKick then
      minKick ← tempKick
      if maxKick >= minKick then
        neglect ← false
      end if
      break //end while
    else
      neglect ← false
      break //end while
    end if
  else
    tempKick ← tempKick + increment
  end if
end while

if neglect then
  return //neglect this passing destination
else
  kickForce ← (minKick + maxKick)/2
  return kickForce
end if
```

Function $r_{opponent\_reach}$ did not depend on the kicking force hence it was not included in our algorithm. The values of function $r_{ball\_too\_fast}$ and $r_{out\_of\_field}$ affected the maximum kicking force. Similarly, the values of function $r_{intercepted}$ and $r_{many\_opponents}$ affected the minimum kicking force. The order of calculation for the first three risks could be altered. For $r_{intercepted}$, the actual kicking force calculation was quite complex and computation intensive. In our algorithm, we decided to use an approximation by slightly increasing the value of the minimum kicking force until the value of $r_{intercepted}$ was no longer exceeding the risk constraint. The final kicking force was an average between the minimum and the maximum kicking force.

There were some modifications to the minimum and the maximum kicking force before the consideration of $r_{intercepted}$. The modifications were to round values to integer values (the type of data remained float). These adjustments were totally for our experiment, which are discussed in Section 4.

4. EXPERIMENTAL METHOD AND RESULT

Our experiment had two phases. Phase 1 was carried out to find a suitable value for risk constraints. Phase 2 was

performed in order to compare the ball passing performance using fixed ball destination with the ball passing performance using unfixed ball destination.

4.1 Phase 1: Finding The Values Of Risk Constraints

We carried out phase 1 by setting up tournament competitions. The first tournament had 128 soccer teams. To prevent a lucky win, each match was played 10 games. Each game lasted 10 minutes. Teams scored points using the rule based on that of Emirates Cup (a pre-season football tournament hosted by English Premier League club Arsenal). We awarded a point for a goal difference advantage. After 10 games, a losing team was immediately eliminated. All teams were given the same configurations except risk constraints. Each team was assigned risk constraints with random values. The tournament winner was likely the team with the best risk constraints. As described in Section 3.1, risk constraints should not be too high or too low. Therefore we had to limit the range of the generated random values.

From initial runs of the experiment, we analyzed that $r_{many\_opponents}$ and $r_{out\_of\_field}$ had considerably less impact than other risk functions. As a result, we decided to fix their risk constraint values to 0.5 and -0.5 respectively. For other risks, the range of random values for risk constraints was from -0.5 to 1. Hence there were three random numbers (out of five risk constraints) to be generated for each soccer team. Based on intial runs of the experiment, we found that more negative values discouraged the AI from passing the ball up the field. To avoid this problem, we defined two heuristics for random number generation. If all three numbers were negative, they would be generated again. If two numbers were negative, they must not be less than -0.3, otherwise the numbers would be generated again, with their range limited from -0.3 to 0 instead of from -0.5 to 1.

Our search space was real number from -0.5 to 1. However, our tournament match only generated 128 teams. Therefore the random values assigned to those team might not cover the entire search space. To compensate for such problem, we had to scale the search space down. We did this by diving our entire search space into slots. Each slot covered 5% of the random range. Therefore our search space was divided into 20 slots. Each slot had the size of 0.075. Each random number was generated from the boundary values of the slots. The winner from this first tournament was regarded as a coarse scale winner.

We then arranged the second tournament. But this time, we used risk constaints of the coarse scale winner to limit the possible range of values generated for the new 128 teams. First, all risk constraints of the teams in the second tournament took their values from those of the first tournament winner. Then, for each value, we varied it by generating a random number between the chosen value and the value one slot away from it. Hence there was a new search space created from each of the first tournament winner's value. We then divided each new search space into slots, each slot covered 5% of the search space range, similar to what we did during the first tournament. The range of random values in the second tournament is shown in Figure 2.

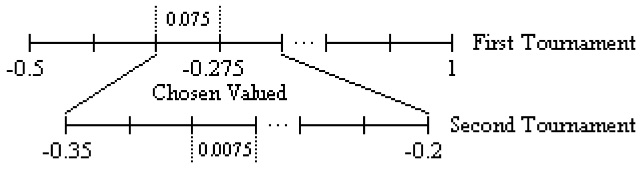


Figure 2: Search Space and Random Slots for Each Tournament

In Figure 2, the chosen value from the first tournament winner was -0.275. Each slot in the first tournament had the size of 0.075 so the value that should be generated was between (-0.275 - 0.075) and (-0.275 + 0.075), which were -0.35 and -0.2. The new search space was between those values. In this example, the range of random values was (-0.2 - (-0.35)) = 0.15. Each slot generated for this range had the size of 5% of the search space range, which was 0.0075.

This second tournament allowed us to generate good teams based on the first tournament winner. However, it was possible that the risk constraints of the first tournament winner were at their local minima. Consequently, 30% of random values were still generated using the same methodology as the first tournament.

The winner from this second tournament was regarded as the refined winner level 1. It was possible to run more tournaments with more detailed random values. We found that when we obtained the refined winner level 4, its performance during the tournament was no longer better than the performance of the refined winner level 3. Therefore our experiments were performed using refined winner level 3. We generated four refined winners at level 3. Risk constraint values from the four teams are shown in Table 1. All teams had the same values for $r_{many\_opponents}$ and $r_{out\_of\_field}$ because they were fixed at 0.5 and -0.5 respectively.

Table 1: Risk Constraints of Refined Winner Level 3

Team	$r_{opponent\_reach}$	$r_{intercepted}$	$r_{ball\_too\_fast}$
1	-0.169875	0.14535	-0.23445
2	0.834075	0.27705	-0.178425
3	0.005025	0.0567	-0.233625
4	0.0714	0.06735	-0.23175

The best risk constraints were found by setting up a league competition between the four teams. The constraint values from the league winner would be used in phase 2 of our experiment. We calculated points like in the tournament competitions. But we increased the number of play rounds for each match to 50 rounds to be certain of the winner. The

result is shown in Table 2. Note that GA stands for 'goal against', goal scored by opponent teams.

Table 2: League Competition Result

Team	Win	Lose	Draw	Goal	GA	Point
1	63	43	44	159	125	267
3	51	43	56	148	129	228
4	50	50	50	132	141	200
2	41	69	40	179	223	163

4.2 Phase 2: Fixed Vs Unfixed Ball Destination Speed

We used the league winner as a standard team for our simulation. We had two teams, one using unfixed ball destination speed and the other using fixed ball destination, compete for 100 games. The result is shown in Table 3. Dribbling was also regarded as passing because it used the same decision process. A risky pass was defined as a successful pass (or dribble) very near an opponent player. The receiving team-mate might lose the ball possession afterwards. There were two slightly different implementions of the team that used unfixed ball destination speed. The only difference was the step of rounding the kicking force to integer value before considering the last risk. The result in Table 3 (under column Unfixed I) was from the team that had this modification. The result in Table 4 (under column Unfixed II) was from the team that did not have the modification.

Table 3: Match Result
Between Teams With Unfixed Ball Destination Speed (Using Integer Round Up) and Teams with Fixed Ball Destination Speed

Statistic	Unfixed I	Fixed
Win/Draw/Lose	67/24/9	9/24/67
Goal scored	222	89
Number of passes	25671	29135
Successful passes	24282	27617
Risky passes	770	381
Shots on goal	437	218
Ball possession (%)	56.8	43.2

Table 4: Match Result
Between Teams With Unfixed Ball Destination Speed (Not Using Integer Round Up) and Teams with Fixed Ball Destination Speed

Statistic	Unfixed II	Fixed
Win/Draw/Lose	55/25/20	20/25/55
Goal Scored	163	93
Number of passes	27891	30963
Successful passes	24685	27695
Risky passes	783	398
Shots on goal	368	241
Ball Possession (%)	57.1	42.9

The result clearly showed that the team using unfixed ball destination speed was superior regardless of the modification. The team that utilized unfixed ball destination speed had less passes but more shots and more goals, implying that it could create a goalscoring chance with less number of playing steps. The ball could be passed to a good position which caused a teammate to easily penetrate the opponent defense.

In Table 3, the rate of successful pass from both teams were around 95%. This showed that both teams had similar passing accuracy when faced each other. We were more interested in the rate of scoring, calculated by dividing the number of goals scored and the number of shots on target. The rate of scoring for the team with unfixed ball destination was 50.8% and the rate of scoring for the team with fixed ball destination was 40.83%. Both teams used the same shooting algorithm. The result indicated that the AI with unfixed ball destination speed managed to shoot the ball from good positions more frequently. This supported our assumption that utilizing unfixed ball destination speed increased the number of good passing destinations. The AI with unfixed ball destination speed executed more risky passes, meaning it was more prepared to take risks in order to pass the ball to good positions.

The performance comparison between two implementations of the team with unfixed ball destination speed (we shall call them Unfixed I and Unfixed II according to Table 3 and 4) is shown in Table 5. We also include the difference (in percentage) between values obtained from each implementation and values obtained from its opposing team (which employed unfixed ball destination speed).

Table 5: Unfixed I and Unfixed II Performance Comparison

Statistic	Unfixed I	Unfixed II
Win (%)	67	55
Goals scored	222	163
Goals scored Diff	+133	+70
Successful passes (%)	94.6	88.5
Successful passes Diff (%)	-0.2	-0.9
Risky Passes (%)	3	2.8
Risky Passes Diff (%)	+1.7	+1.5
Rate of scoring (%)	50.8	44.3
Rate of scoring Diff (%)	+10	+5.7
Ball possession (%)	56.8	57.1
Ball possession Diff (%)	+13.7	+14.2

From Table 5, the implementation that utilized the rounding of values to integers (Unfixed I) performed noticeably better. It had a higher percentage of win, goal difference, rate of scoring and rate of scoring difference. For other values, the two implementations performed similarly. Although the implementation with the rounding of values to integers had

more pass successes than the implementation without the rounding of values, it did not have more pass successes when compared to the opponent. The result indicated that the AI had high successful passes. But to exactly know how good it was, it was useful to compare it with real professional team. In World Cup 2010, the winner and runner-up were Spain and Netherlands. Their passing accuracy percentages were 89.6% and 88.4% respectively (guardian.co.uk website). This showed that the AI had very high passing accuracy. In addition to the data from Table 5, computational time and the number of no pass were used to compare the two unfixed ball destination implementation. A no pass happened when the AI was not able to find even one passing destination because all positions failed their risk constraints. In our simulation, the AI dealt with such situation by dribbling away from the nearest opponent player. A computational time was the average time (in milliseconds) our system used from the start of a ball passing evaluation until the AI could find the optimum passing destination. StopWatch class from C# .NET was employed for this measurement. Low values for the number of no pass indicated that more positions were being considered when passing, hence better play qualities were more likely to take place. Low values for the computational time simply indicated faster speeds. We collected these statistics from the matches used in Table 3 and Table 4. The collected data is shown in Table 6.

Table 6: Number of No Pass and Computational Time (ms)

Statistic	Games from Table 3		Games from Table 4	
	Unfixed I	Fixed	Unfixed II	Fixed
No. of no pass	2602	5806	6067	8387
Average Computational time	31.13	30.95	34.78	29.5

From Table 6, both teams (Unfixed I and Unfixed II) had less numbers of no pass than their opponents. This confirmed that teams with unfixed ball destination speed were really able to increase the number of passing destinations in their decision making process. When comparing Unfixed I and Unfixed II, Unfixed I is better, both in raw value and the value relative to its opponent.

For the computational time, Unfixed I and Unfixed II had more computation time than their opponents, which was normal since they performed more complex algorithm. When comparing Unfixed I and Unfixed II, Unfixed I is clearly better. On average, it was only 0.18 ms slower than its opponent. On the other hand, Unfixed II was, on average, 5.28 ms slower than its opponent. This showed that Unfixed I was about 29.33 times faster than Unfixed II. The reason for this might be because when rounding a kicking force to integer value, the range between the minimum and the maximum kicking forces decreased, which also decreased the number of incrementation needed when considered function $r_{\text{intercepted}}$ in our algorithm. Another possible reason was that the system could calculate our algorithm using integer values much faster than using floating point values. Note that the computational time in Table 6 was not collected when a no pass took place. A no pass event did not result in Pareto optimality evaluation so including it would give incorrect information because we are only interested in the performance when using Pareto optimality evaluation.

5. CONCLUSION AND FUTURE WORK

We proposed an improved ball passing algorithm for the Pareto optimum passing AI in soccer games. In our approach, the ball speed could be varied to allow previously high risk positions to be included in the Pareto selection. The results showed that varying the ball speed really improved scoring chances. The rounding of values to integers before calculating the value of the last risk function also helped to improve the quality of play and the execution time. The simulation executed at an average of 60-70 frames per second (fps) on the system with Window XP SP3, Core2 Duo 2.0 GHz, Geforce 8600M GS and 3.0 GB of RAM. For future experiments, we intend to enhance our AI by implementing a long ball passing decision and allowing our soccer players to head the ball. These skills are very crucial to a soccer player. They create more varieties in game play and make the game much more exciting.

ACKNOWLEDGMENT

This research is sponsored by Ratchadapiseksompoj Funds, Chulalongkorn University.

REFERENCES

Buckland, M. 2005. “Programming Game AI by Example”, ISBN: 1556220782

Bustamante, C.; Flores, C.; and Garrido, L. 2007. “A Physics Model for the RoboCup 3D Soccer Simulation” In *Proceedings of the 2007 spring simulation multiconference - Volume 2* (SpringSim '07), Vol. 2. Society for Computer Simulation International, San Diego, CA, USA, 151-158.

Coello Coello, Carlos A.; Lamont, Gary B.; and Van Veldhuizen, David A. 2007. “Evolutionary Algorithms for Solving Multi-Objective Problems”, Springer (2007), ISBN: 978-0-387-33254-3

Kyrylov, V. 2007. “Balancing Gains, Risks, Costs, and Real-Time Constraints in the Ball Passing Algorithm for the Robotic Soccer” In: Lakemeyer, G. et al. (Eds.) *RoboCup 2006: Robot Soccer World Cup X.* LNCS (LNAI), vol. 4434. Springer Berlin / Heidelberg (2007), 304-313.

Noda, I.; Matsubara, H.; Hiraki; and K., Frank, I. 1998. “Soccer server: A tool for research on multiagent systems” *Applied Artificial Intelligence 12* (1998), 233-250.

Reis, L.P. and Lau, N. 2001. “FC Portugal Team Description: RoboCup 2000 Simulation League Champion” In: Stone, P.,

Balch, T., Kraetzschmar, G.K. (eds.) *RoboCup 2000*. LNCS (LNAI), vol. 2019, Springer, Heidelberg (2001), 29-40.

Stone, P. and McAllester, D. 2001. "An Architecture for Action Selection in Robotic Soccer" In: *Proceedings AGENTS'01, 5th International Conference on Autonomous Agents*, May 28-June 1, 2001, Montreal, Quebec, Canada (2001), 316-323.

Xu Yuan and Tan Yingzi. 2008. "Rational Passing Decision Based on Region for the Robotic Soccer" In: Visser U. et al. (Eds.) *RoboCup 2007: Robot Soccer World Cup XI*. LNCS (LNAI), Vol. 5001. Springer Berlin / Heidelberg (2008), 238-245.

guardian.co.uk website (accessed date 23/01/2011) http://www.guardian.co.uk/football/datablog/2010/jun/17/opta-world-cup-2010-data

Unity3D Game Engine. 2010. (accessed date 23/01/2011) http://unity3d.com/

STRATEGY GAMING

DIFFICULTY BALANCING IN REAL-TIME STRATEGY GAMING SESSION USING RESOURCE PRODUCTION ADJUSTMENT

Piyapoj Kasempakdeepong
Vishnu Kotrajaras
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Payathai Road, Patumwan,
Bangkok 10330, Thailand
E-mail: picmee29@gmail.com, vishnu@cp.eng.chula.ac.th, ajarntoe@gmail.com

KEYWORDS
Real-time Strategy Games, Artificial Intelligence, Dynamic Scripting, Score Function

ABSTRACT

Most researches in the real-time strategy games focus on Artificial Intelligence that beats a human opposition. However, each human player has different skill. An Artificial Intelligence with certain skill level causes skillful players to become bored of the game because it is too easy for them. At the same time, new players may find the game too difficult. Consequently, players may stop playing the game altogether. This paper presents a novel Artificial Intelligence that uses unit production to adjust appropriate game difficulty for players. We use the open-source real-time strategy engine, Spring, in our experiment. Score functions are generated automatically to obtain our fitness function for dynamic scripting and top culling. Our adjustment, unlike other researches, applies within a playing session instead of between sessions. The result shows that a gaming session can be adjusted to suit players' abilities without any need to tune the behavioral script of the game.

INTRODUCTION

Players vary in skill. A game artificial intelligence (AI) with scripted behavior is therefore not suitable for all players. Skilled players may be bored with a given game while new players struggle to beat the game's AI. This can result in poor sales. Hence, a game AI that adjusts its difficulty level when facing different players needs to be developed.

The AI for real-time strategy games has been researched continuously during the past few years. Marc Ponsen and Pieter Spronck created an AI for real-time strategy games (Ponsen et al. 2005; Spronck and Ponsen 2008) by using dynamic scripting in an open-source game, Wargus. They divided a game session into different states according to constructed set of buildings in the game. Each state was given a specific rulebase. Rules were selected from the rulebase according to their weights, which changed according to their performance. The AI improved in quality using this technique, but the technique was not designed to regulate the difficulty to suit each player. For the AI that adjusted its skill according to its human opponent, three different enhancements to dynamic scripting were proposed for Computer RolePlaying Games (CRPGs) (Spronck et al. 2006). The enhancements included high-fitness penalization, weight clipping, and top culling. Dynamic scripting which incorporated top culling was shown to be the best compared with high-fitness penalization and weight clipping (Spronck et al. 2004a). Still, difficulty adjustments were only performed after each playing session, not during the session. Potisartra and Kotrajaras proposed a methodology for difficulty scaling during a game session (Potisartra and Kotrajaras 2010), having their AI estimate its opponent's skill based on movements of units in the past (and in possible near future), and choose to execute actions that evenly match the opponent's skill. However, the estimation was only possible for turn-based strategy games, where scores for each movement of each unit were defined clearly.

This paper presents an alternative form of dynamic scripting that can be used with real-time strategy games to adjust the game's difficulty level during play. We used dynamic scripting with top culling. However, our dynamic scripting rules, instead of controlling the behavior of each unit, managed the rate of unit production. In addition, score functions were used to find the value of fitness function during play to allow real-time difficulty adjustment. Even players who used a highly optimized tactics would still be challenged by our AI.

The outline of the paper is as follows. The next section presents how dynamic scripting, top culling and score function operate. Then our unit production adjustment is presented, followed by experimental results and discussions. The final section concludes and establishes future directions.

RELATED WORK

Dynamic Scripting

The dynamic scripting technique was used to develop an AI that adapted successfully to changing circumstances for CRPGs (Spronck et al. 2004b). The methodology involved

generating a set of rules, using the method of random rule selection from a given rulebase. A rule with higher weight had a higher probability for getting selected. After the end of a game, a fitness function was used to evaluate the performance of selected rules and change the weight of each rule. The weights of a set of rules that contributed to a winning game would be increased; on the other hand, the weights of a set of rules that led to losing would be decreased. However, after any weight change was completed, the value of each weight must still be in the range between w_{min} and w_{max}. The remaining rules were updated in order to retain the total value of the weights in the rulebase.

The maximum weight value w_{max} represented the highest level of optimization allowed for a learned tactic. A high value of w_{max} permitted the weights to rise higher. As a result, most effective rules would be picked frequently. The script would therefore get close to its optimal form. On the contrary, a low value of w_{max} limited the increase of weight, diversifying selected rules in the scripts.

Top Culing

Top culling was shown to be quite effective for adjusting the difficulty level of dynamic scripting to suit players (Spronck et al. 2004a). It prevented rules with high values of weight from being selected. It achieved this by reducing the value of w_{max} by a constant amount (w_{dec}) when the AI won and increasing the value of w_{max} by a constant amount (w_{inc}) when the AI lost. However, the value of w_{max} must not be lower than the initial weight value w_{init}. The weight of a rule was allowed to go above w_{max}, but rules with weight higher than w_{max} would not be chosen for script generation. Therefore, when the game AI won frequently, the weight of most effective rules would be higher than w_{max}, making such rules unusable. The AI then consequently used weaker tactics. On the other hand, when the game AI lost frequently, rules with high weight would become selectable again, allowing stronger tactics to be used.

From figure 1, after w_{max} is given a new value, rule 2 has its weight exceeding w_{max}. Therefore, in the next script selection round, rule 2 will not be available.

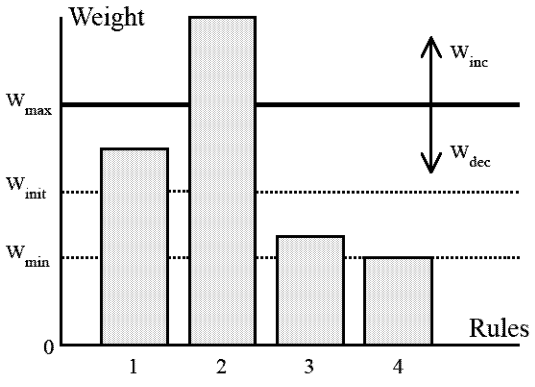


Figure 1: Top Culling Process

Score Function

Bakkes's score function (Bakkes et al. 2007a; Bakkes et al. 2007b; Bakkes et al. 2007c; Bakkes and Spronck 2008) was shown to be capable of effectively evaluating the status of the game; both when the game ended (absolute prediction) and at any point of the game (relative prediction). Bakkes's score function is defined as

$$v(p) = w_p \left(\sum_u w_u \left(Cu_0 - \frac{Cu_1}{R} \right) \right) + (1 - w_p) \left(\sum_{r \in D} w_r \left(\frac{Or_1}{Rr_1} - \frac{Or_0}{Rr_0} \right) \right) \quad (1)$$

From Equation 1, $v(p)$ is the score function at a specific time in phase p.

w_p is a free parameter that indicates the importance of each term of score function in each phase of the game. Bakkes used Gradient Descent Optimization Algorithm to find w_p.

w_u is the weight of each unit type that can be learned by Temporal Difference Learning.

Cu_0 is the number of unit type u belonging to the game AI.

Cu_1 is the number of visible unit type u belonging to the opponent.

R is a fraction of the environment that is visible to the game AI.

w_r is the weight of radius r that is obtained from the experiment.

Or_1 is the number of visible units of the game AI within distance r around the opponent's commander.

Or_0 is the number of visible units of the opponent within distance r around the game AI's commander.

Rr_1 is a fraction of the environment that is visible to the opponent commander.

Rr_0 is a fraction of the environment that is visible to the game AI's commander.

A player who would win the game was expected to have a positive score. A player who would lose the game was expected to have a negative score. Bakkes used Temporal Difference Learning (Beal and Smith 1997) to search for the weight of each unit (w_u).

Score Function was a key component for measuring the state of a game at any specific time. We applied Bakkes's score function and used it to construct our fitness function in order to measure the performance of the game AI during play. The value of the fitness function was then used as feedback for our dynamic scripting.

EXPERIMENTAL SETUP

Spring

We used Spring, an open-source real-time strategy engine, to set up our experiment. A Spring mod, Balanced Annihilation version 7.19 (Figure 2), was chosen as our testing game. Its gameplay involved resource collection, units and buildings creation. The objective of the game was

to destroy the opponent's commander. The selected map for our experiment was SmallDivide, which was a symmetrical map with no water area.

Game AI

We modified an open-source AI for Spring called E323, developed by F. Huizinga and slogic, so that we were able to control the production of units by using dynamic scripting. For other behaviors such as organizing defense, moving units to attack or patrol, the default script for E323 was used.

For our dynamic scripting, our rules consisted of the various production rates for each unit. Each rule was assigned its initial weight so that it could be used effectively without additional learning time.

Figure 2: Balanced Annihilation Mod in Spring Engine

Fitness function was used for measuring the performance of the script. In Spronck's original dynamic scripting proposal, a fitness function was used as part of weight adjustment function to adjust the weight of selected rules in the script at the end each game. However, we wanted our dynamic scripting to be able to adjust the weight throughout one gaming session. Therefore a script performance measurement methodology was required. We used Bakkes's score functions, which predicted the result of each game and also showed the performance of the AI at any given time, as part of our fitness function.

Our fitness function at time t is defined as

$$F_t = V_t - V_{t-1} \qquad (2)$$

From Equation 2, F_t is the fitness function at time t,
V_t is the score function at time t,
V_{t-1} is the score function at time $t-1$,
t is the time that the dynamic scripting operates.

If the fitness function returned a positive value, it indicated that the selected rules in the script worked effectively. In such case, our weight adjustment functions would increase the weight of the selected rules in the script, so that the rules had more probability of being selected next time the dynamic scripting was called. If the fitness function returned a negative value, it indicated that the selected rules in the script did not work well. Our weight adjustment functions would therefore reduce the weight of the selected rules in the script, so that the rules had less probability of being selected next time the dynamic scripting was called.

The modified E323 attempted to adjust its behavior to win. In order to adjust E323 to play evenly with players, top culling technique was applied. The result from using top culling was recorded in table 1.

Figure 3 shows the overall process of Our AI. Stars (*) in the figure indicates our main contributions.

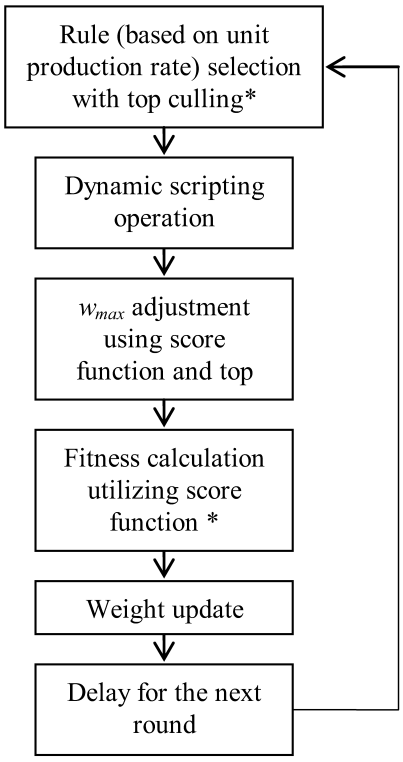


Figure 3: The Overall Operation of Our AI

Difficulty Scaling

Top culling technique, similar to dynamic scripting, was originally used at the end of a playing session to adjust w_{max}. We re-applied the technique so that it could be used throughout the playing session. At any time, the score function at that time was analyzed. If its value was positive, meaning the AI was playing better than the player, top culling would decrease the value of w_{max} by w_{dec} percent, making effective rules less likely to be selected in the next round of dynamic scripting processing. The AI would therefore degrade in its quality of play. In contrast, if the value of the score function was negative, implying the AI was playing worse than the player, top culling would increase the value of w_{max} by w_{inc} percent, making effective rules more likely to be selected in the next round of dynamic

scripting processing. The AI would therefore appear smarter.

Weight Setup

We used Temporal Difference Learning to find the weight of each unit type. We added an external module in Spring to extract required data from training games. The data was collected every 90 game cycles (3 seconds), consistent with the updated frequency of E323. We had E323 fight against E323 for 700 matches and E323 fight against AAI (another AI that was implemented for Spring) for 700 matches. From the matches, 2 sets of weight of each unit type (one set for fighting against itself and the other set for fighting against AAI) were obtained.

Gradient Descent Optimization Algorithm was used to find the weight for each term of the score function in each phase of the game (w_p) by selecting the w_p that maximized the proportion of succesful predictions over the total number of predictions in phase p of all games. We used the proportion in our calculation in order to compensate for different game time so that longer games did not have an advantage.

RESULT & DISCUSSION

Performance Evaluation

To evaluate the performance of our game AI, we compared the result from having it fight against regular E323 and regular AAI. We also set E323 against itself, AAI against itself, and E323 against AAI. Each pair of battle consisted of 200 matches.

We had two types of measurements, average game time and win/lose percentage. If the average game time our AI used when playing against E323 (or AAI) was greater than the average game time E323 used to fight against E323 (or AAI), then the game AI could be regarded as having the same quality compared to E323. The same argument applied when AAI was used as a friendly team.

If the win/lose percentage of our game AI when playing against a regular AI was similar to the win/lose percentage of E323 (or AAI) when playing against that regular AI, we regarded our AI as having at least the same quality as E323 (or AAI)

Table 1: Result of the measurement

Friendly team	Opponent team	Game time (s)	win	loss	Win/lose (percent)
E323	E323	775.50	85	115	42.5/57.5
Adapted game AI	E323	1414.38	112	88	56/44
AAI	AAI	1149.08	66	134	33/67
Adapted game AI	AAI	1513.80	111	89	55.5/44.5
E323	AAI	718.76	200	0	100/0

Discussion

From table 1, considering the average game time, E323 took 775.50 seconds on average to defeat E323. When the friendly AI was changed to our AI, the average game session lasted up to 1.8238 times of the original. AAI took 1149.08 seconds on average to defeat another AAI. When we changed the friendly AI to our AI, the average game time became 1.3104 times of the original value. Having longer game sessions indicated that our AI was able to ward off its opponent better, while not rushing to win at the same time. This provided even matches we needed.

For the win/lose percentage, we compared 42.5/57.5 (E323 battling E323) with 56/44 (our AI battling E323) and we compared 33/67 (AAI battling AAI) with 55.5/44.5 (our AI battling AAI). It was clear that our AI gave the win/lose ratio closer to 50:50. This result further confirmed that dynamic scripting on unit production rate with real-time score function was effectively usable as a methodology for dynamic difficulty adjustment.

E323 always beat AAI. This was due to E323 having better behavioral scripts. Our AI was able to fight both types of AI to long drawn-out matches with good win/lose ratio. This indicated that our AI was effective with both weak and strong players.

CONCLUSION & FUTURE WORK

In this paper we proposed a game AI that used unit production to adjust difficulty level to suit its opponent. We measured the quality of the proposed AI by comparing average game time and win/lose ratio with matches that used regular AI. From the longer average game time and the closer to 50:50 win/lose ratio, we concluded that our proposed AI was able to scale its difficulty level according to its opponent.

For future work, our dynamic scripting needs to be modified so that it produces appropriate units in each phase of the game. Currently, weak units can still be produced in late game, which does not make much sense. In order to correctly produce certain units in certain phases, weights of rules for producing weaker units in a phase needs to be transferable to rules for producing stronger but similar units in the next phase. With such successful weight transfer, we hope to achieve even better average game time and win/lose ratio.

REFERENCES

Bakkes, S., P. Spronck, and J. V. D. Herik. 2007a. "Phase-dependent Evaluation in RTS Games". In M. M. Dastani and E. D. Jong (Eds.). *Proceedings of the 19th Belgian-Dutch Conference on Artificial Intelligence*, 3-10.

Bakkes, S., P. Kerbusch, P. Spronck, and J. V. D. Herik. 2007b. "Automatically Evaluating the Status of an RTS Game". In M. V. Someren, S. Katrenko and P. Adriaans (Eds.). *Proceedings of the Annual Belgian-Dutch Machine Learning Conference*, 143-144.

Bakkes, S., P. Kerbusch, P. Spronck, and J. V. D. Herik. 2007c. "Predicting Success in an Imperfect-Information Game". In J. V. D. Herik, J. Uiterwijk, M. Winands and M. Schadd (Eds.). *Proceedings of the Computer Games Workshop*, 219-230.

Bakkes S. and P. Spronck. 2008. *Automatically Generating a Score Function for Strategy Games*. AI Game Programming Wisdom 4, 647-658.

Beal, D. and M. Smith. 1997. "Learning Piece Values using Temporal Differences". *International Computer Chess Association Journal*, No. 20, 147-151.

Ponsen, M., H. Muñoz-Avila, P. Spronckl, and D. W. Aha. 2005. "Automatically Acquiring Domain Knowledge for Adaptive Game AI Using Evolutionary Learning". *The Twentieth National Conference on Artificial Intelligence*, 1535-1540.

Potisartra K. and V. Kotrajaras. 2010. "An Evenly Matches Opponent AI in Turn-based Strategy Games". In *Proceedings of 2010 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT 2010),* 42-45.

Spronck P. and M. Ponsen. 2008. *Automatic Generation of Strategies*. AI Game Programming Wisdom 4, 659-670.

Spronck P., M. Ponsen, I. Sprinkhuizen-Kuyper, and E. Postma. 2006. *Adaptive Game AI with Dynamic Scripting*. Machine Learning, vol. 63, No. 3, 217-248.

Spronck, P., I. G. Sprinkhuizen-Kuyper, and E. O. Postma. 2004a. "Difficulty Scaling of Game AI". In A. E. Rhalibi and D. V. Welden (Eds.). *Proceedings of the GAME ON 5th International Conference on Intelligent Games and Simulation*, 33-37.

Spronck, P., I. G. Sprinkhuizen-Kuyper, and E.O. Postma. 2004b. "Online Adaptation of Game Opponent AI with Dynamic Scripting". In N. E. Gough and Q. H. Mehdi (Eds.). *International Journal of Intelligent Games and Simulation*, vol. 3, 45-53.

STRATEGIES TO SOLVE A 4x4x3 DOMINEERING GAME

Jonathan Hurtado
Department of Computer Science
Digipen Institute of Technology
9931 Willows Rd NE, Redmond, Washington 98052, USA
E-mail: jonathan.hurtado@digipen.edu

KEYWORDS
Combinatorial games, Domineering, three-player games

ABSTRACT

3D Domineering is a three-player variant of the classic two-player combinatorial game, Domineering. Researchers have so far solved a x b x c Domineering games where $a + b + c < 10$ and $a, b, c \geq 2$. In this paper, we solve a 4x4x3 Domineering game. In fact, we show that it is not possible for any of the three players to have a winning strategy.

INTRODUCTION

Domineering

Domineering is a two-player combinatorial game that is played on a grid board of any size (ex. 3x3, 4x6, 10x4, etc.), although it is usually square to be fair to both players. Each player can place a domino piece that covers two adjacent spaces on the board. Both players take turns placing a domino on the board (the domino cannot overlap other domino pieces) until one player is unable to place any more pieces, which results in a loss for that player. One player, referred to as left player or L, can only place pieces vertically on the board. The other player, referred to as right player or R, can only place pieces horizontally on the board. Figure 1 shows a brief example of a Domineering game.

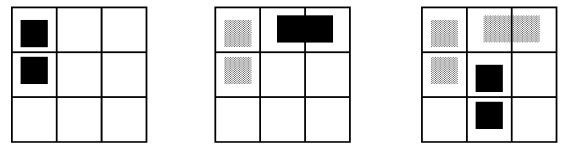


Figure 1: Example of a Domineering Game

3D Domineering

3D Domineering is a three-player variant of the Domineering game. It is played on a rectangular solid board similar to a Rubik's cube whose edges are parallel to the x, y, and z axes. The three players have pieces that occupy two adjacent cubes of the board and take turns in a cyclical order placing them in the 3D board. One player is limited to placing pieces along the x-axis of the board, another player is limited to placing pieces along the y-axis of the board, and the remaining player is limited to placing pieces along the z-axis of the board. If a player is unable to place any more pieces in the board at his turn, then that player is eliminated from the game. The remaining two players will continue to play until one is unable to place any more pieces in the board, resulting in a loss for that player.

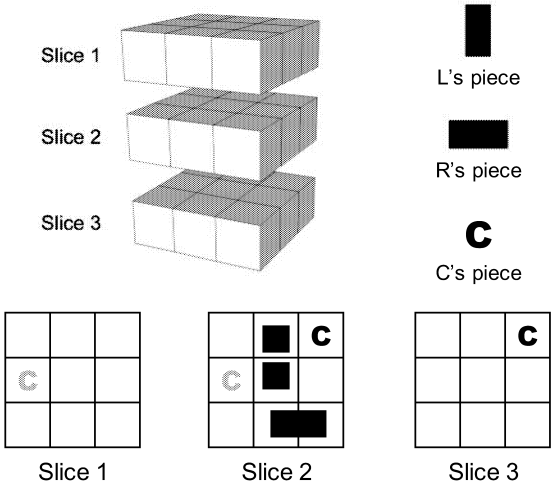


Figure 2: Example of a 3x3x3 Domineering Game

Figure 2 shows an example of a 3x3x3 Domineering game. We marked the three players as L (left), R (right), and C (center). Let's suppose that the turn order for this game is C-L-R. C plays a piece that traverses Slices 1 & 2 on the board. Then L and R play their pieces on Slice 2. C follows up by playing a piece that traverses Slices 2 & 3. All three players continue to take turns until two of the players are no longer able to place any more pieces.

Solving a Domineering Game

A Domineering game is considered *solved* if one can find which player has a winning strategy for each turn order. A winning strategy is a set of moves that will always guarantee a win for one player regardless of the other player's actions (this is also known as *forcing a win*).

Brueker, Uiterwijk and van den Herik created the program DOMI to solve m x n boards where $2 \leq m \leq 8$ and $m \leq n \leq 9$ (Brueker et al. 2000). Bullock later improved on their research by developing a search application called Obsequi that could not only solve those games faster, but also solve

a 10 x 10 board, which is currently the largest solved Domineering game (Bullock 2002).

The addition of a third player in 3D Domineering makes analysis of those games more complex than 2D Domineering. This is because you need to consider all possible moves made by three players instead of two, and there is a chance that no player can force a win. Straffin introduced a concept called "decision by player", where one player who is unable to win a three-player game can affect which of the other two players can win with his next move (Straffin 1985). Thus, it is possible that one player cannot force a win no matter what he does if one of his opponents plays a move that causes the other opponent to win. A 3D Domineering game is therefore considered solved when, for each turn order, one can either identify the winning strategy for a player or prove that no player has a winning strategy.

Alessandro Cincotti used an extensive search algorithm to solve 3D (a x b x c) Domineering games where $a + b + c < 10$ and $a, b, c \geq 2$ (Cincotti 2007). Attempts to analyze games where $a + b + c \geq 10$ is difficult to complete because exhaustively searching all of the game's possible moves is computationally expensive.

We address that issue by formulating strategies that can significantly reduce the number of moves to search in a 3D Domineering game. Specifically, we found strategies for two players that can prevent the third player from winning, regardless of the turn order and the third player's actions. Selecting specific moves for the two players, instead of considering all their possible moves, makes it faster to search all possible moves and determine if any of the three players has a winning strategy for each turn order.

We will demonstrate how these strategies prove that a 3x3x3 Domineering game is a Q game, a game where no player can force a win in any turn order. We will then expand on those strategies to show that a 4x4x3 game is also a Q game.

SOLVING A 3x3x3 DOMINEERING GAME

Alessandro Cincotti had already solved the 3x3x3 Domineering game to be a Q game, but we will still present our strategies because they will serve as the foundation to solve a 4x4x3 game.

Proposition: *In a 3x3x3 game of Domineering, it is impossible for one player to force a win if the other two players form an alliance and collude to stop him.*

Since a 3x3x3 game is symmetrical on all sides, if C cannot force a win if L and R collude to stop C, then it also means that L cannot force a win if R and C collude to stop L, and R cannot force a win if L and C collude to stop R. Thus, demonstrating the above proposition will show that no player can force a win in a 3x3x3 Domineering game.

Let's say that L and R form an alliance against C. There exists a two-part strategy that L and R can follow that will prevent C from winning regardless of whether C plays first, second, or third. To be clear, a win for C in a three-player Domineering game means that C still has at least one legal move left after both L and R have no more legal moves.

Notice in Figure 2 that when C places a piece, it traverses either Slices 1 and 2 or Slices 2 and 3. Thus, if all the spaces in Slice 2 are occupied, then C cannot make any more moves. Therefore, the first part of the blocking strategy that L and R should adopt to prevent C from winning is as follows:

Strategy 1: *L & R should place as many pieces as possible on Slice 2 first before placing any pieces on Slices 1 or 3.*

Since L and R are going to be working together to stop C, we should consider this scenario as a special two-player game between C and the L & R alliance. This means that if C is unable to play any legal moves before both L and R are eliminated, then the alliance wins. Furthermore, the alliance has an advantage over C where the alliance can continue to play even if one of its members is unable to place any more pieces.

We will demonstrate an example of the L & R alliance's strategy to cover as much of Slice 2 as possible using the following L-R-C game.

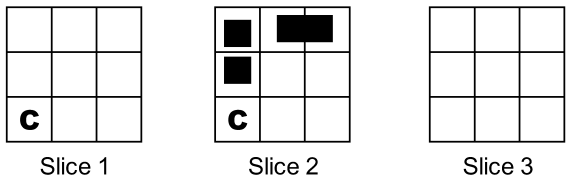


Figure 3: A sample 3x3x3 L-R-C game after one turn

At the start of a 3x3x3 Domineering game, there are nine available spaces in Slice 2. After one complete turn where all three players place their first move in Figure 3, there are four spaces left in Slice 2. One can see in Figure 3 that only one alliance member can move on Slice 2 in his next turn. The other alliance member must move on either Slice 1 or Slice 3, as demonstrated in Figure 4.

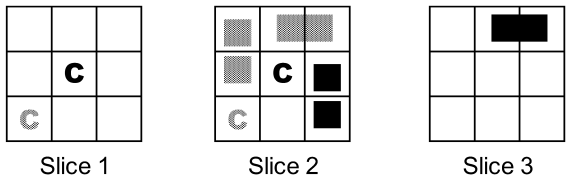


Figure 4: A sample 3x3x3 L-R-C game after two turns

After two complete turns, there is one space left in Slice 2 of Figure 4 (meaning that C has one more move left), but neither alliance member can place any more pieces in Slice

2. However, the alliance can still block C with the second part of its blocking strategy.

Strategy 2: *When there are no more available moves in Slice 2 for the alliance, L and R can cooperatively block C's remaining moves by placing pieces above and below C's available spaces on Slice 2.*

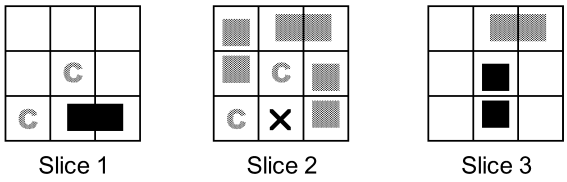


Figure 5: A sample 3x3x3 L-R-C game after three turns

In Figure 5, the alliance's third moves block C from playing on the last available space on Slice 2 (indicated by the x). C has no more available moves, which results in a loss for C.

While this only covers one specific 3x3x3 game, we found that Strategies 1 and 2 are effective in all possible 3x3x3 games (Hurtado 2010). For example, in games where C moves last (L-R-C and R-L-C), the alliance can limit C to a maximum of one space in Slice 2 after two complete turns.

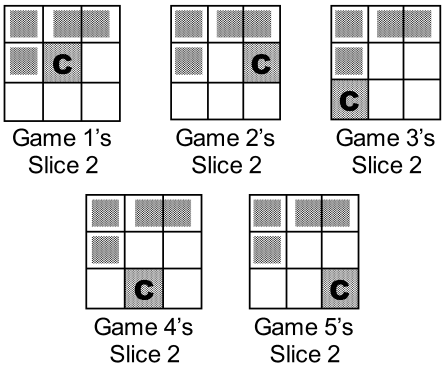


Figure 6: Potential Slice 2s of L-R-C & R-L-C 3x3x3 Games after one complete turn

After L and R make their first moves, there are five possible spaces that C can occupy in Slice 2 during his first turn, as shown by C's shaded moves in the five games of Figure 6. In the upper-left diagram, both alliance members can move in Slice 2 in their next turn, which eliminates C. In the other diagrams, only one alliance member can move in the remaining spaces of Slice 2 in his next turn, which leaves two spaces left. When C occupies one of those spaces with his second move, there is just one space left in Slice 2, which the alliance can cover using Strategy 2 in their next turn, resulting in a loss for C.

In games where C goes second (L-C-R and R-C-L), the alliance can limit C to one space on Slice 2 after C's second turn.

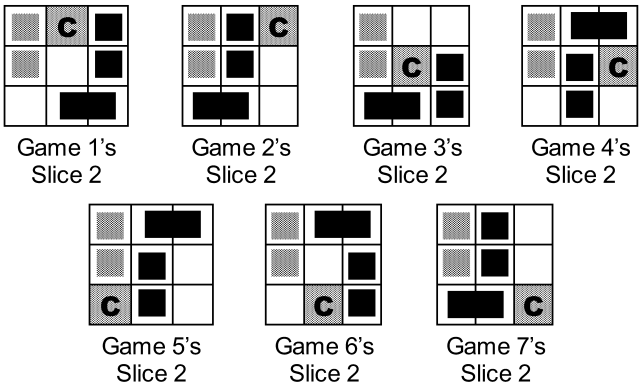


Figure 7: Potential Slice 2s of L-C-R (& R-C-L) 3x3x3 Games before C's second move

After L makes his first move on the upper-left corner of Slice 2, C can occupy seven spaces on Slice 2 in his first move, as shown by C's shaded moves in the seven games of Figure 7. In each game of Figure 7, Slice 2 is left with two spaces when L & R move after C's first turn. When C makes his second move in each game, Slice 2 is left with one space, which the alliance can cooperatively block using Strategy 2 in its next turn, eliminating C from the game. If you rotated all the games in Figure 7 90°, then the examples in Figure 7 would also cover games where R moves first.

For games where C goes first (C-L-R and C-R-L), the alliance can once again limit C to one space on Slice 2 after two turns unless C plays the right moves. We found that C's best opening move is on a space that is neither corner nor center. This is because it allows him to prevent the alliance from covering any more spaces on Slice 2 with a second move demonstrated in Figure 8.

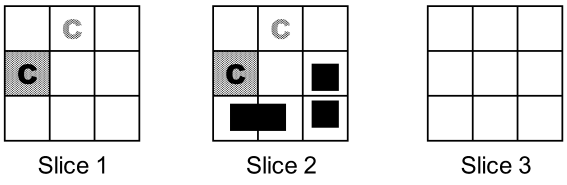


Figure 8: C prevents the alliance from blocking Slice 2 and reserves a move

As long as C plays the moves indicated in Figure 8, he will have three available moves in Slice 2 after his second turn instead of one. Furthermore, placing both his moves on Slices 1 & 2 allows C to protect a move in the upper-left corner of the board from any blocking attempts by L or R. This tactic is referred to as *reserving a space*.

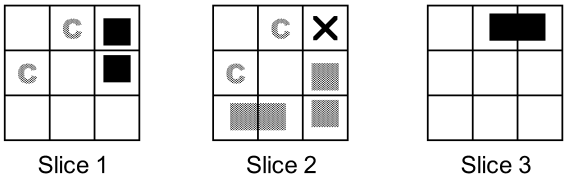


Figure 9: L & R block one of C's moves

Since the alliance is unable to play on Slice 2, they must perform Strategy 2 and cooperatively block the upper-right space of the board, as seen in Figure 9 (C's blocked move is indicated by the x). This leaves C with just two moves left after two complete turns. C could play on his reserved space (upper-left corner) in his next turn, but this allows the alliance to cooperatively block the center space in their next turn, eliminating C. Therefore, C's better move is to play on the center space.

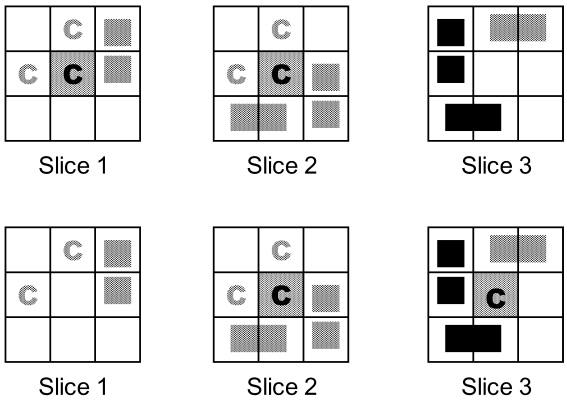


Figure 10: Both games show C playing on the center space

Figure 10 shows the two moves C could play that cover the board's center space (which are highlighted in gray). The alliance is unable to cooperatively block C's last move since it is reserved, so it instead plays on Slice 3 in both games. After C plays his last move on the upper-left corner, one can see in the two games of Figure 10 that both L and R have still one more move left in the game. Thus, C loses because he is eliminated before both L and R.

Regardless of turn order and where C plays, the alliance is always able to prevent C from winning as long as it follows the winning strategy. Thus, we have reconfirmed what Cincotti already solved (a 3x3x3 game is a *Q* game) using Strategies 1 & 2. In the next section, we expand on those strategies to solve a 4x4x3 Domineering game.

SOLVING A 4x4x3 DOMINEERING GAME – PART 1

A 4x4x3 game is not symmetric on all sides, so this affects our analysis of where L, R, and C play in that game's board. We previously mentioned that the edges of a 3D Domineering board are parallel to the x-, y-, and z-axes. In this paper, we define that R, L, and C play along the x-, y-, and z-axes respectively.

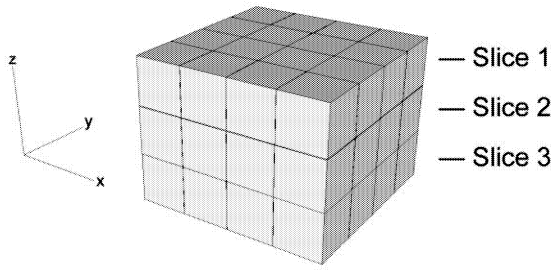


Figure 11: Establishing the x-, y-, and z- axes of a 4x4x3 game

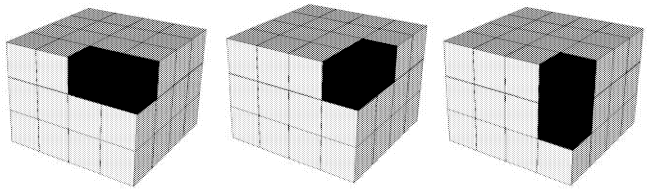


Figure 12: R's, L's, and C's moves (highlighted in dark gray) are respectively parallel to the board's x-, y-, and z-axes

We establish these definitions to make it clear where R, L, and C are playing when presenting how to solve a 4x4x3 game (in an *a* x *b* x *c* game, R plays on *a*, L plays on *b*, and C plays on *c*).

We will first show that the alliance can prevent C from winning a 4x4x3 Domineering game if C plays on the 3 component. We will then show that the alliance can prevent C from winning a 4x4x3 Domineering game if C plays on the second 4 component (which will be subsequently referred to as a 4x3x4 game).

Demonstrating that C cannot force a win in a 4x4x3 game if L and R collude against him, regardless of turn order or whether C plays on the 3 component or the second 4 component, proves that no player has a winning strategy in a 4x4x3 game. This is because if C cannot force a win if L and R team up against C, it also means that L cannot force a win if R and C team up against L, and R cannot force a win if L and C team up against R.

Proposition: C *is unable to force a win in a 4x4x3 game, where C plays on the 3 component, if the L & R alliance teams up against him.*

The strategies that L & R can use to do this are similar to the strategies that prevented C from winning a 3x3x3 game.

Strategy 1: *L & R should place as many pieces as possible on Slice 2 first before placing any pieces on Slices 1 or 3.*

Strategy 2: *When there are no more available moves in Slice 2 for the alliance, L and R can cooperatively block C's remaining moves by placing pieces above and below C's available spaces on Slice 2.*

After analyzing all possible moves that C can do in each turn order, we have found that the alliance can reduce the number of C's available spaces in Slice 2 to a maximum of three

spaces within three turns when following Strategy 1 (Hurtado 2010). In games where C goes last, the alliance can limit C to one space on Slice 2 within three turns. In games where C goes first or second, on the other hand, the alliance can only limit C to three spaces on Slice 2 within three turns. In this section, we'll focus on the latter games.

Figure 13 shows an example of a 4x4x3 game where C has three moves left after three turns.

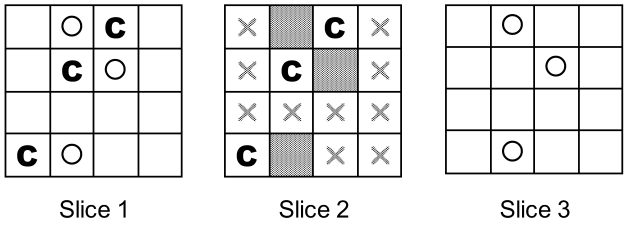


Figure 13: A potential 4x4x3 game with C having three spaces left (indicated by the shaded squares and Os)

The shaded squares in Slice 2 represent C's available spaces on that slice. The Os in Slices 1 and 3 represent where the other half of C's move could go when he plays on an available space in Slice 2. The gray x's in Slice 2 represent potential pieces places by L and R when executing Strategy 1. Although this diagram represents a specific 4x4x3 game, we claim that no matter how you configure C's three moves and C's three available spaces in Slice 2 in a 4x4x3 game, the alliance is still able to cooperatively block at least one of C's available moves using Strategy 2 and successfully prevent C from winning.

Here's how the alliance is able to block at least one of C's available moves. There exists no configuration on a 4x4 slice where C's three pieces will protect all three of his potential moves. This is because in order for C to protect one of his moves from the alliance on either Slice 1 or Slice 3, he needs at least two pieces.

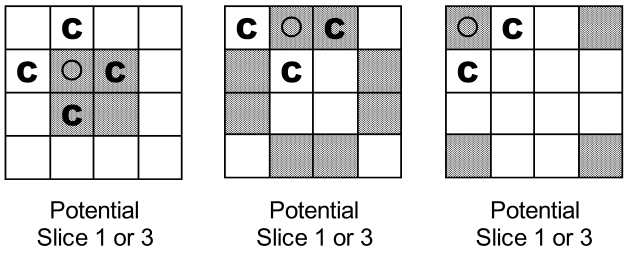


Figure 14: A demonstration of how C can protect his potential moves (Os)

Figure 14 shows how many C pieces are needed to protect a C potential move (O) depending on its location. The first diagram of Figure 14 demonstrates that four C pieces are needed to protect potential C moves in center spaces (which are shaded). This scenario, however, is not possible because C has only moved three times before this scenario occurred (there would only be three C pieces, not four).

Figure 14's second diagram reveals that three C pieces are needed to protect Os that are on spaces that are neither corner nor center (which are shaded). Since all of C's three pieces are protecting only one potential space, this leaves the other potential moves unprotected, which can be covered by an alliance member.

Figure 14's third diagram is the best case for C because it shows that only two C pieces are needed to protect Os that are on corner spaces (which are shaded). However, if we were to place another O on that board, C could not protect it with his third piece because we just showed that C needs at least two pieces to protect a potential move.

If C cannot protect two potential moves with his three pieces, he certainly cannot protect three potential moves with his three pieces. Therefore, we can confidently state that the alliance can cooperatively block at least one of C's potential moves

Why is this important? We stated that C has three moves left after three turns. If C occupies one of them in his next turn, and the alliance cooperatively block one of C's remaining moves in its next turn, then C will only have one move left by the fifth turn. There would still be plenty of spaces in Slices 1 and 3 for the alliance to move even after C plays on his last available space, which is a loss for C. In games where C goes last, it's worse for C because C only has one available space in Slice 2 after three turns, which the alliance can cooperatively block in their next turn.

Therefore, we have shown that for all possible games on a 4x4x3 board, where C plays on the 3 component, C cannot force a win if the alliance teams up against him and uses Strategies 1 & 2. However, we have not completely solved the 4x4x3 game yet. We still need to show that the alliance can prevent C from winning even if you rotate the 4x4x3 board so that C plays on the second 4 component (4x3x4).

SOLVING A 4x4x3 DOMINEERING GAME – PART 2

To complete our demonstration that a 4x4x3 game is solved to be a *Q* game, we must demonstrate the following:

Proposition: *C is unable to force a win in a 4x3x4 game, where C plays on the 4 component, if the L & R alliance teams up against him.*

Similar to the 3x3x3 and 4x4x3 (where C plays on the 3 component) games, the alliance has a two-part winning strategy to prevent C from winning. Since C is able to move on four slices in a 4x3x4 game instead of three, the alliance will need to cover two slices of the game board in order to prevent C from making any more moves. Thus, the first part of the alliance's winning strategy in 4x3x4 games is:

Strategy 1: *Cover as much of Slices 1 & 3 as possible for the duration of the game until L and R are unable to move on either slice.*

It is crucial that the alliance commits to playing on either Slices 1 or 3 until they cannot move on those slices anymore. Deciding which of the two slices to play will depend on where C last played.

C will play on Slices *j* and *j+1* in his turns, where *j* = 1, 2, or 3. If *j* = 1 or 3, the alliance should play on the same Slice *j*. If *j* = 2, then the alliance should play on Slice 3. However, if an alliance member is unable to play on one of the slices in the Slices 1 & 3 pair, then he should move on the other slice in the pair if a move is available. For example, let's say C played on Slices 1 and 2. According to our strategy, the alliance should move on Slice 1 on its next turn. If L or R is unable to move on Slice 1, then that alliance member should play on Slice 3 if a move is available. If an alliance member is unable to play on either Slice 1 or Slice 3, then he must follow the second part of the alliance's winning strategy.

Strategy 2: *The alliance must cover the corresponding squares above and below any remaining moves for C to block him from playing them. If the alliance is unable to cooperatively block an available move for C, then both players should strive to either block available C moves that only need one piece to block or partially block an available C move.*

There are two cases to consider when the alliance starts the second part of its winning strategy. The first case is when both alliance members are unable to move on Slices 1 and 3 in their next turn. The second case is when only one alliance member is unable to move on Slices 1 and 3 in his next turn. We will start the analysis of Strategy 2 with the first case.

Case 1: *Both alliance members can no longer play on Slices 1 and 3*

If both alliance members can no longer move on Slices 1 and 3 in their next turn, then they should cooperatively block an available C space in their next turn if the opportunity exists. The following diagram shows an example of this strategy.

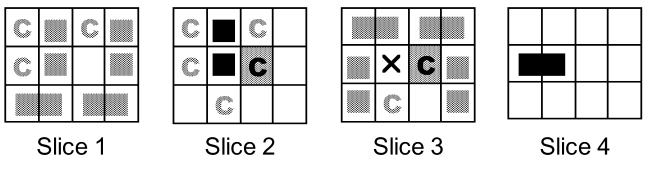


Figure 15: L & R cooperatively block a C potential move

In this sample 4x3x4 game, the alliance plays moves above and below an available C move to block it (indicated by the x) after C's fifth turn (indicated by the shaded square).

However, the alliance needs to be aware which of C's potential moves they cooperatively block. The following diagrams demonstrate why.

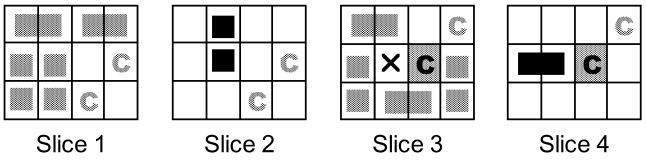


Figure 16: L & R cooperatively block one C move

In Figure 16's sample 4x3x4 game, the moves L & R made after C's fourth move (indicated by the shaded square) are not an optimal winning strategy for the alliance. This is because there is another pair of moves that the alliance can make that can block two of C's available moves (indicated by the two x's in Figure 17).

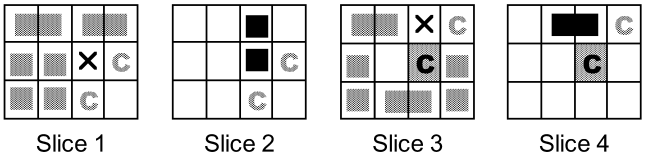


Figure 17: L & R cooperatively block two C moves

Thus, the following summarizes the alliance's optimal winning strategy for Case 1.

- *If there are cooperative moves that can block more than one of C's available moves, then the alliance should play them.*
- *If that is not possible, then the alliance should play a pair of cooperative moves in their next turns that can block one of C's available moves.*
- *If cooperative blocking moves are not available, then both players should block different available moves for C in their next turns.*
- *If none of the above applies to an alliance member, then the member will need to pick a valid move. A valid move in this case is one that would not prevent the other alliance member from playing a piece that would have blocked one of C's available moves.*

We continue the analysis of Strategy 2 with the second case.

Case 2: *Only one alliance member can no longer play on Slices 1 and 3*

The one alliance member who cannot play on Slice 1 or Slice 3 should look for moves that can block one of C's available spaces using one piece. The following diagram shows an example of this strategy.

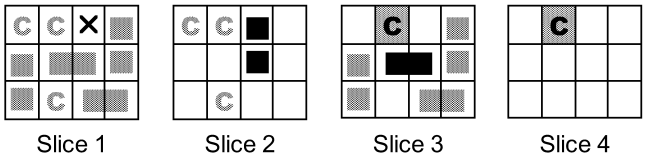


Figure 18: A sample 4x3x4 game (L-C-R)

After C's fourth turn (indicated by the shaded square) in Figure 18's sample game, R plays on Slice 3. L is unable to play on Slice 1 or Slice 3, so he places a move on Slice 2 that blocks C (indicated by the x).

This alliance member, however, must be careful if he happens to go before the other alliance member who can play on Slice 1 or 3. Here's why.

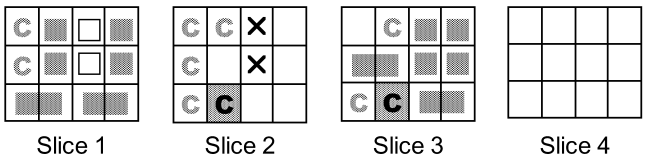


Figure 19: A sample 4x3x4 game (L-C-R)

It is R's turn after C's fifth turn in Figure 19's sample game and he can neither play on Slices 1 or 3. L still has a move left on Slice 1 (indicated by the open squares) that would block two of C's spaces in Slice 2 (indicated by the x's). If R were to make a move that would overlap one of those x's, it would be a wasted move for the alliance since R could have blocked another C move that would not have been blocked by L's move, as indicated in Figure 20.

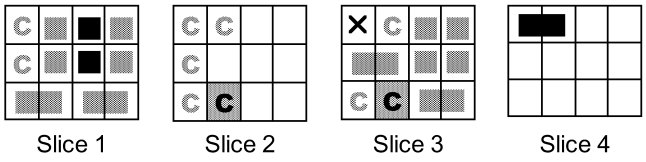


Figure 20: C is eliminated in this game

The following summarizes the alliance's winning strategy for Case 2.

If only one alliance member cannot play on Slice 1 or Slice 3 in his next turn, he should play a move that can fully block an available C space using only one piece. If that member goes first, however, he must not block an available C space that will also be blocked by the other alliance member's next move on a slice that is in the Slices 1 & 3 pair.

We developed an application that simulated all possible 4x3x4 games that followed the strategies described in this section for two of the players (in this case, L & R) and considered all possible moves made by the third player (C). The application generated six text files from the 4x3x4 game simulation, one for each turn order (C-L-R, C-R-L, L-R-C, R-L-C, L-C-R, R-C-L). The following is an excerpt from one of those text files:

```
Program started at: 1284406810
Dimensions of Domineering board: 4x3x4
Turn order for all games in this simulation: left, right, center

Game Number: 1 + (10000000 x 0)
  Winner: left/right
    left:    0,1,0  3,1,0  2,1,0  3,0,2  1,1,2  2,1,1
    right:   0,0,0  2,0,0  0,0,2  2,2,2  1,0,3  2,1,3
    center:  1,1,0  1,2,0  0,1,1  0,2,1  2,0,1
    # of Remaining Moves: - left: 5 - right: 5 - center: 0

...

Game Number: 114377 + (10000000 x 0)
  Winner: left/right
    left:    0,1,0  0,1,2  1,1,2  3,1,0  1,1,0  0,0,1
    right:   0,0,0  0,0,2  2,0,2  2,0,0  2,1,1  0,2,1
    center:  3,2,2  3,1,2  2,2,2  2,1,2  2,2,0
    # of Remaining Moves: - left: 5 - right: 7 - center: 0

Program finished at: 1284406831
```

Figure 21: An excerpt from the output of the 4x3x4 L-R-C game simulation

The first lines of the output indicate the program's start time, the dimensions of the board, and the turn order for all games simulated by the application. The simulation stops analyzing a game when either C wins or has no more available moves (regardless of whose turn it is). It outputs how many times the simulation stopped analyzing a game (indicated in the Game Number line), whether C won or not, and what moves each player made for that specific game (indicated by a three-number coordinate that marks the first half of the player's move). It also writes the number of remaining moves left for each player. When the application finishes traversing through all possible games, it writes the time before it quits.

In each turn order output file, performing a search for "Winner: center" yielded no results. This is how we determined that *there are no games where C won when the alliance performed its winning strategy*. If you were to take a random game from any of the output files and played the moves indicated in that game, it will show that the alliance performed their winning strategy to prevent C from winning.

Table 1 outlines how long the application took to simulate all possible games for each turn order and how many games it simulated.

Turn Order	Duration	Number of games
CLR	251 seconds	1,415,738
CRL	259 seconds	1,455,637
LCR	81 seconds	453,729
RCL	83 seconds	458,325
LRC	21 seconds	114,377
RLC	21 seconds	116,839
Total:	**716 seconds**	**4,014,645**

Table 1: 4x3x4 3D Domineering simulations using two-part winning strategy

We also modified the application to simulate all possible 3x3x3 C-L-R games (where none of the players use any type of strategy) to compare results with the 4x3x4 simulations in Table 1.

Turn Order	Duration	Number of Games
3x3x3 – CLR	Approx. 8 hours	1,034,224,512

Table 2: 3x3x3 3D Domineering simulations

You may have noticed that we simulated 3x3x3 games in Table 2 instead of 4x3x4 games. This is because simulating all possible 4x3x4 games would have taken the application an extremely long time to finish, as there are a substantial number of possible moves to traverse.

Furthermore, the simulations in Table 2 are not actually complete simulations—they only considered all possible moves made by the player whose turn was next at each position. This differs from a complete analysis of a 3D Domineering game because one must consider all possible moves that can be made by *all three players* at each game's position, regardless of whose turn it is. Even with a 3x3x3 board, this would have created a very large number of moves for the application to traverse, which is why we limited the simulations to only consider moves for one player at each position. Nonetheless, we were able to estimate the total number of moves of a completely analyzed 3x3x3 game using the Number of Games from Table 2, which we found to be approximately 201,280,264,483 (Hurtado 2010).

We can make the case that the total number of possible games for a 4x3x4 game would be significantly higher than the number of games for a 3x3x3 game. We make this point because the total time and number of games from our 4x3x4 simulations would clearly be significantly less than a complete analysis of a 4x3x4 game.

CONCLUSION

We have demonstrated in the last two sections that C cannot force a win in a 4x4x3 game if L and R collude against him, regardless of turn order or whether C plays on the 3 component or the second 4 component. Therefore, we have shown that the 4x4x3 is solved to be a *Q* game, as no player can force a win in a 4x4x3 game, regardless of board orientation or turn order.

FUTURE WORK

Solving a 4x4x3 Domineering game using the strategies outlined in the previous sections opens the possibility of solving 3D Domineering boards of larger sizes without having to analyze all possible moves for those games. We previously determined that the alliance's optimal strategy to prevent C from winning 3x3x3 and 4x4x3 (where C plays on the 3 component) games is to cover as much of the middle slice as possible. One consideration for future work is to investigate if there is a limit to how large a game of m x n x 3 (where $m > 4$, $n > 4$, and C plays on the 3 component) can be where blocking the middle slice remains an effective strategy for the alliance to block C.

Another consideration for future work is to test if having the alliance play on alternating slices is an effective strategy to prevent C from winning regardless of how many slices C can play on. We found that when C plays on four slices in a 4x3x4 game, the strategy to block C is to play on alternating slices (Slices 1 & 3). If C played on five slices (for example, 5x5x5), can the alliance prevent C from winning by playing on Slices 2 & 4? And if C played on six slices (for example, 6x6x6), can the alliance prevent C from winning if they played on Slices 1, 3, and 5? What if the board size was skewed to C's favor, such as a 3x3x8 board, with C playing on the 8 component? Would playing on alternate slices still be enough for the alliance to prevent C from winning? How big does the board have to skew in C's favor before C can force a win despite the other two players teaming up against him? These are questions that can be explored with further research.

REFERENCES

Breuker, D.M.; J.W.H.M. Uiterwijk; H.J. van den Herik. 2000. "Solving 8 x 8 Domineering." *Theoretical Computer Science*, 230, 195-206.

Bullock, N. 2002. "Domineering: Solving Large Combinatorial Search Spaces." University of Alberta.

Cincotti, A. 2008. "Three-Player Domineering." *Proceedings of World Academy of Science, Engineering, and Technology*, 36, December, 92-95.

Hurtado, J. 2010. "Finding Strategies to Solve a 4x4x3 Domineering Game." Digipen Institute of Technology.

Straffin, Jr., P. 1985. "Three Person Winner-Take-All Games with McCarthy's Revenge Rule." *The College Mathematics Journal*, 16, November, 386-394.

BIOGRAPHY

JONATHAN HURTADO was born in Cali, Colombia and obtained his undergraduate degree in Computer Science from New York University in 2001. After working as a web developer for several years, he pursued a graduate degree in Computer Science from the Digipen Institute of Technology, which he obtained in 2010. He is currently working in Digipen's Singapore campus.

GRAPHICS

Real-Time Object-Space Edge Detection using OpenCL

Dwight House
Master of Science in Computer Science
dwighthouse@gmail.com

Dr. Xin Li
Dean of Faculty, Director of Education
DigiPen Institute of Technology
xli@digipen.edu

Abstract

At its most basic, object-space edge detection iterates through all polygonal edges in each mesh to find those edges that satisfy one or more edge tests. Those that do are expanded and rendered, while the remainders are ignored. These 3D edges, and their resulting accuracy and customizability, set object-space methods apart from all other categories of edge detection. The speed and memory limitations of iterating through all polygonal edges in each mesh each frame has inspired optimization research.

In this paper, we explore methods to calculate object-space edges utilizing programmable GPU technologies, including OpenCL. The OpenCL methods explored allow for a significant reduction in calculation quantity. Some also provide a reduction in rendering artifacts and memory usage over previous GPU techniques. Unfortunately, most uses of OpenCL for edge detection results in slower performance than shader-based techniques, though variations and optimizations may reduce this disadvantage in the future.

Keywords: Non-photorealistic Rendering, Edge Detection, Edge Drawing

Background

Edge detection and rendering is an important non-photorealistic rendering technique. Rendered edges can be used for a multitude of purposes including object differentiation, structural enhancement, highlighting, and blur-based anti-aliasing. They also are a required component of several graphical styles such as toon rendering and sketchy rendering.

Edge Types

There are several types of edges. Each type requires different detection tests, and may not be detectable with a given method.

Contour: A polygon edge that connects two polygons, one front-facing and the other back-facing.
Crease: A polygon edge that connects two polygons that are within some user-defined angular distance of each other.
Boundary: A polygon edge that forms the side for only one polygon.
Intersection: A collision of two polygons such that the line formed along the intersection is a polygon edge of only one or neither of the colliding polygons.
Marked: A polygon edge flagged to always be rendered as an edge.

For the purposes of this paper, we ignore further discussion of intersection edges. Intersection edges are not detectable with object-space methods.

Edge Detection Method Categories

Object-space edge detection refers to one category of edge detection. The other categories are Hardware, Image-space, and Miscellaneous.

Hardware: In these methods, edges are not detected, but directly rendered as a byproduct of some other operation, or series of operations. These methods are very fast, available on almost all hardware, and do not require mesh preprocessing or additional memory. However, they typically only render contours and they lack customizability.
Image-space: In this method, two rendering passes are used. In the first, scene data, like depth and normal information, is rendered and stored to the GPU temporarily. In the second pass, the stored data is used with image-processing techniques to determine areas of rapid change, which generally correspond to edges. This method has constant-speed edge detection, because the speed of the second pass is dependent on the number of pixels in the viewport, not the number of objects rendered. It can also detect edges easily that other methods cannot. However, the edge accuracy is lacking and the edge thickness is inconsistent. The image-space, like the hardware methods, does not produce edges that are readily customizable.
Object-space: In these methods, some or all of the polygon edges of each mesh are tested for their drawability. Those edges that pass the test are then expanded into quads or other structures, where they are rendered like any other geometry. Those polygon edges not drawable are discarded. Since the edge detecting occurs in 3D space, the results are more accurate. The output is 3D geometry, so the edges can be customized to a great degree, including texturing, animating, and shading. However, the number of tests and rendering operations for each edge causes object-space edge detection to be the slowest form of edge detection.

Miscellaneous: These methods share no direct similarities, though in terms of their positive and negative qualities, they resemble hardware methods.

In this paper, we only focus on object-space edge detection, as it is the method used by this paper's OpenCL-based edge detection.

OpenCL

OpenCL, which stands for Open Computing Language, creates a standardized interface for computational tasks on a variety of devices. It is specifically focused on data-parallel tasks, which require a single set of relatively simple operations be performed on massive quantities of nearly atomic data. This is essentially the form of calculation found in GPU rasterization. Not surprisingly, GPUs are often the preferred device for utilizing OpenCL. Like vertex shaders, an OpenCL program (called a kernel) can access data stored on the GPU and perform operations on it before creating output. However, OpenCL is more free in what it can do and what it can access. Though more versatile, without careful planning an OpenCL operation can take far longer than the equivalent operation implemented as a shader.

Previous Work

Edge detection algorithms are very old and well documented. However, the paper by Markosian et al. [Markosian et al. 97] represents some of the first work in real-time edge detection. They stored edge adjacency information for each polygon edge. Contour edges tend to form loops around a mesh, so when a contour edge was found, they recursively performed the contour edge test on the connecting edges first. This method tends to detect the longest, and therefore most significant, edges with a minimal number of random tests. Additionally, a small portion of contour edges detected each frame were also stored for the next frame as starting points for the next round of edge tests. Without sudden movements, a significant number of contour edges remain the same from frame to frame. By checking only a very small percentage of all edges, they found a fivefold increase in the rendering speed over testing all polygon edges individually. Of course, some contour edges could be missed entirely from frame to frame, possibly resulting in flickering.

Gooch et al. [Gooch et al. 99] described a method where they stored edges' normal arc on a sphere surrounding the object. Groups of similar arcs, in gauss map format, were stored hierarchically so that groups of edges could quickly be deemed all back-facing or all front-facing. A plane was placed at the origin of the sphere and then aligned perpendicular with the view vector. Edges whose arc intersect the plane are contour edges. This technique allowed contour edge detection to be sped up by 1.3 times for their S. Crank mesh and 5.1 times for a sphere. Unfortunately, this technique only works well under orthographic projection.

In a similar idea, Sander et al. [Sander et al. 00] created a hierarchical search tree of polygons. At each node, they created anchored cones that represented the maximum range of the normals possessed by vertices in the node. This information can be used to quickly determine that no contour edges are possible for whole sets of nodes without testing individual edges.

Jeff Lander [Lander 01] documented the optimization of ignoring edges that have co-planar adjacent polygons. Flat planes only generate drawable contour edges on their outside edges, not their internal edges, and they lack the angular difference between adjacent polygons to generate crease edges. During the preprocessing step, an additional test checks for co-planar adjacent polygons. If found, the edge is not added to the list of edges to test. If a mesh was constructed using primarily quads, this optimization can reduce the number of edge tests significantly: over 20% in the case of the Utah Teapot.

Other imaginative edge storage and detection methods exist. Aaron Hertzmann and Dennis Zorin [Hertzmann and Zorin 00] described a method of using 4D dual surfaces to determine the contour edges with curve-plane intersections. Tom Hall [Hall 03] created a modification of Markosian et al.'s technique by focusing almost exclusively on tracking contour changes from frame to frame. By looking at adjacent edges to previously found edges and noticing the relative camera change, he was able to significantly reduce the number of edges tested. This method worked especially well with highly tessellated meshes.

Finally, Morgan McGuire and John F. Hughes [McGuire and Hughes 04] detailed a method to shift the entirety of the edge detection and rendering to the graphics card via programmable shaders. Copies of adjacency, vertex, and

normal information were stored in vertex buffer arrays and accessed within the vertex shader. If the edge was found drawable, the degenerate duplicate vertices were turned into screen-aligned quads. Otherwise, they were shifted behind the camera, where they would be clipped during the rendering process. McGuire and Hughes also took a critical look at how the thick edge gaps could be filled effectively. Their paper formed the basis for OpenCL edge detection research.

Hardware Determined Feature Edges

We focused heavily on improving the object-space method described by Morgan McGuire and John F. Hughes [McGuire and Hughes 04]. To provide context and ease understanding, we detail here McGuire and Hughes' work as it relates to our contributions.

McGuire and Hughes' goal was to transfer all detection and rendering of edges to the GPU, gaining the speed of parallel calculation and removing the bottleneck of geometry data transfer from the CPU to the GPU. Parallel computation and the technology of the time forced them to test every polygon edge independently, unlike some of the object-space optimizations. As a result, a large amount of GPU memory is required.

The method used four render passes:

Mesh pass: Render mesh normally at a slight backward offset
Edge pass: Render drawable edges as expanded quads or lines
Cap pass (first side): Render the first side cap of each drawable edge
Cap pass (second side): Render the second side cap of each drawable edge

Data Structure

McGuire and Hughes created an edge mesh data structure that contained four edge vertices for each unique polygon edge. Each edge vertex contained values required for edge detection and generation. Those that can be represented geometrically are shown in Figure 1.

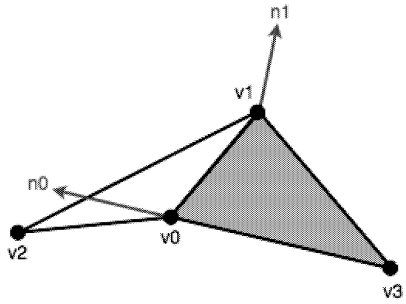


Figure 1: The important geometric values for each edge in McGuire and Hughes' method: v0, v1, v2, and v3 are vertices on two polygons that share an edge. n0 and n1 are the vertex normals for v0 and v1, respectively. v3 may not exist in the case of a boundary edge.

The values in each edge vertex are:

v0: first vertex in the edge
v1: second vertex in the edge
v2: final vertex that, with v0 and v1, makes the first polygon
v3: final vertex that, with v0 and v1, makes the second polygon
n0: v0's normal vector
n1: v1's normal vector
r: random scalar used in texture parameterization
i: scalar from 0 to 3 that differentiates duplicates in the edge mesh

The r parameter is ignored for this paper. The i parameter is used to pick what value to output after an edge is detected drawable. If the edge is a boundary edge, there is no v3 vertex. In that case, v3 is set equal to v0.

For each polygon edge, all of the above values are obtained from the mesh, except for i. The newly formed edge vertex is duplicated three times. Each of the now four edge vertices with the same data are given a different, ordered i value from 0 to 3. All four of these edge vertices for each polygon edge is stored with all other edge vertices in an edge mesh. Finally, the edge mesh is copied into vertex buffers on the GPU for later use. This preprocessing step is quite expensive, but once complete, all edge detection and rendering can be accomplished by the GPU with no special data transfer whatsoever.

The user needs some way of referencing the edge mesh buffers on the GPU. Generating an index array buffer containing sequential numbers from 0 to (4E - 1), where E is the number of unique polygon edges, allows the data to be referenced and rendered with a single render call. Furthermore, other index array buffers can be created to reference only the first three, or first two, i values of each set of four edge vertices. Such index array buffers are useful for rendering caps and thin edges.

Edge Detection

Contour edges can be detected by finding the dot product of both adjacent polygons' face normals with the view vector. If the sign of the two values is different, the polygon edge is a contour. Crease edges can be detected using the face normals of the adjacent polygons as well. Since the face normals' angular distance is inversely proportional to the angular "sharpness" of the two adjacent polygons, polygon edges are found to be creases if the dot product of the two face normals is less than the negative cosine of the user-defined angle. This will detect crease edges that are any sharper (smaller angular distance between adjacent polygons) than the user-defined angle. Finally, boundary and marked edges are detected by checking v0 and v3 for equality. The detection tests are described algorithmically below.

Contour: [(DotProduct(Na, V) * DotProduct(Nb, V)) < 0]
Crease: [DotProduct(Na, Nb) < -cos(θ)]
Boundary and Marked: [v0 == v3]

Where Na and Nb are the face normals of the two adjacent polygons, V is the view vector, and θ is the user-defined angle.

Once an edge is detected, one of several rendering methods can be used to create a viewable edge feature.

Edge Rendering

The user has several choices for what type of edge to generate, including a rasterized line, full quad, or half-quad. For each type of output, the i value is used to determine which of several output vertices should be generated.

Rasterized Line Edge

For rasterized lines, only two duplicates per edge vertex are needed in the edge mesh. If available, it's most efficient to use an index buffer that contains only the indices of the first two edge vertices per set in the edge mesh. When the i value is 0, the vertex shader should output MVP * v0. Otherwise, it should output MVP * v1. MVP is the ModelViewProjection matrix.

McGuire and Hughes create edges, and later caps, in screen-space to ensure a consistent thickness. The screen-space versions of some edge components are needed to accomplish the creation process. Those values are:

s0: v0 in screen-space
s1: v1 in screen-space
m0: n0 in screen-space
m1: n1 in screen-space
p: normalized perpendicular vector to the screen-space edge vector (s1 - s0)

These values are calculated via:

s0
```
vec4 s0 = MVP * vec4(v0.xyz, 1.0);
s0.xy = (s0.xy / s0.w) * vec2(Width, Height);
```

s1
```
vec4 s1 = MVP * vec4(v1.xyz, 1.0);
s1.xy = (s1.xy / s1.w) * vec2(Width, Height);
```

m0
```
vec4 temp = MVP * vec4(v0.xyz + n0.xyz, 1.0);
temp.xy = (temp.xy / temp.w) * vec2(Width, Height);
vec2 m0 = normalize(temp.xy - s0.xy);
```

m1
```
vec4 temp = MVP * vec4(v1.xyz + n1.xyz, 1.0);
temp.xy = (temp.xy / temp.w) * vec2(Width, Height);
vec2 m1 = normalize(temp.xy - s1.xy);
```

p
```
vec2 p = normalize(vec2(s0.y - s1.y, s1.x - s0.x));
```

Where Width and Height are the width and height of the viewport, respectively. The above functions use component-based division and swizzle vector operations. The p vector's length can be modified to adjust the screen-space thickness of rendered edges.

Full Quad Edges

Full quad edges render on both sides of the screen-space edge, as seen in figure 2. On contour edges, the "inner" half would penetrate the mesh itself, which can sometimes lead to artifacts. However, for most marked and boundary edges, and all crease edges that are not also contours, these are the preferred output type. Using the i value, one of four calculations listed below are performed on the input edge vertex to create the output vertex. Note that the output vertices are converted from screen-space back to projection-space, which is the expected output format for vertex shaders.

i = 0 vec4((s0.xy - p.xy) / vec2(Width, Height) * s0.w, s0.zw)

i = 1 vec4((s1.xy - p.xy) / vec2(Width, Height) * s1.w, s1.zw)

i = 2 vec4((s1.xy + p.xy) / vec2(Width, Height) * s1.w, s1.zw)

i = 3 vec4((s0.xy + p.xy) / vec2(Width, Height) * s0.w, s0.zw)

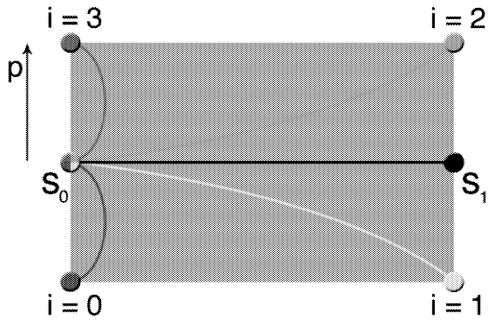


Figure 2: Depending on the i value, the output point is calculated using the two screen-space edge points and the perpendicular vector.

Half-Quad Edges

Half-quad edges are only suitable for contour edge rendering. To ensure that the half-quad is rendered on the "outside" of the mesh, the p value is modified so it always points in the same direction as the m0 vector via the sign function. As with the full quad method, the i value is used to determine which of four output values to create. Figure 3 provides an illustration of the output.

i = 0 vec4(s0.xy / vec2(Width, Height) * s0.w, s0.zw)

i = 1 vec4(s1.xy / vec2(Width, Height) * s1.w, s1.zw)

i = 2 vec4((s1.xy + p.xy * sign(dot(m0, p))) / vec2(Width, Height) * s1.w, s1.zw)

i = 3 vec4((s0.xy + p.xy * sign(dot(m0, p))) / vec2(Width, Height) * s0.w, s0.zw)

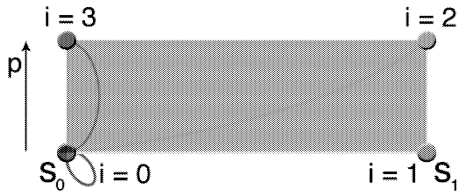


Figure 3: The p vector is modified to point outward, so the edge half-quad will only render on the outside of the mesh.

Non-Drawable Edges

If an edge fails all the edge tests, it is a non-drawable edge. Optimally, these edge vertices would be removed from the pipeline at this point, since they do not create valid output. However, vertex shaders cannot delete vertices. McGuire and Hughes solved this issue by outputting a single vertex for all i values: (0, 0, -1, 1). These vertices will not only be degenerate, but in projection-space they are behind the camera and will be clipped before reaching the fragment shader.

Edge Caps

Edges rendered with a thickness of greater than a few pixels will cause visible gaps to occur at many of the edge connection points, as in the left side of figure 4. McGuire and Hughes accounted for these by rendering half-caps on both ends of each drawable edge, connecting at a point along the screen-space normal of the side in question to create a cap that fills the gap completely (figure 4 on right). The half-caps can be generated from the same edge mesh data as the edge, but they need a different index array buffer. The cap index array buffer needs index values that reference only the first three duplicates of each set of edge vertices.

As mentioned earlier, the two caps are generated in separate render passes: the first deals with the v0 side and the second deals with the v1 side. They are rendered as triangles, rather than quads. The output vertices are described below.

Half-Cap Output On v0 Side

i = 0 vec4(s0.xy / vec2(Width, Height) * s0.w, s0.zw)

i = 1 vec4((s0 + p * sign(dot(m0, p))) / vec2(Width, Height) * s0.w, s0.zw)

i = 2 vec4((s0.xy + m0) / vec2(Width, Height) * s0.w, s0.zw)

Half-Cap Output On v1 Side

i = 0 vec4(s1.xy / vec2(Width, Height) * s1.w, s1.zw)

i = 1 vec4((s1 + p * sign(dot(m1, p))) / vec2(Width, Height) * s1.w, s1.zw)

i = 2 vec4((s1.xy + m1) / vec2(Width, Height) * s1.w, s1.zw)

The p, m0, and m1 vectors should all be scaled by the same value used when scaling the edge if the user desires a specific screen-space thickness.

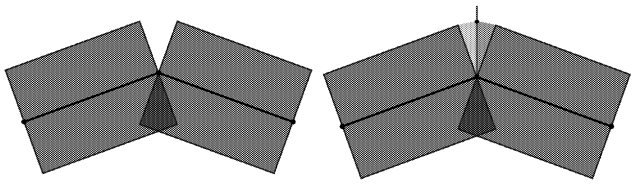


Figure 4: On the left, two thick screen-space edge quads connect to create a visible gap. Using the screen-space projection of the vertex normal they share, the gap can be filled with a cap formed from one half-cap from each edge (one blue, one red), as seen on the right.

Issues

McGuire and Hughes' edge and capping method are highly effective and very fast on modern hardware, but they have a few drawbacks.

Calculation Duplication: Each polygon edge is tested for drawability ten times: four for the edge and three for each half-cap. Once an edge is determined drawable, all the calculations necessary to create the output vertices is also duplicated. Geometry shaders usage was proposed as a solution, since it can output more than one vertex per invocation.
High GPU Memory Usage: The edge mesh also requires at least seventy-two extra floats and four integer values for every polygon edge. All of this information is already on the GPU in the form of vertices, vertex normals, and indices. McGuire and Hughes suggested using data textures to reduce the memory usage by a factor of four.
Incorrect Caps: When the screen-space projection of the vertex normals does not correspond well to the curvature of the edge, caps can be generated on the wrong side of the edge under certain perspectives. McGuire and Hughes suggested rendering caps on both sides of the edge when the error is likely to occur.
Screen-Space Thickened Edges: While it is trivial to scale the thickness of the drawn edges and caps, the use of screen-space scaling makes all edges have the same thickness no matter their distance from the camera. This can cause artifacts and confusion about the distance of the object. Though screen-space thickness may be desired, having a depth-based scaling factor is beneficial for other graphical requirements.

Using OpenCL can help with the first three issues. The last issue is dealt with in the appendix.

Object-Space Edge Detection Using OpenCL

After researching and implementing McGuire and Hughes' method, we attempted to improve upon it. The most extensive research focused on finding a replacement technique that would take advantage of the new capabilities of OpenCL.

Relevant OpenCL Capabilities

OpenCL has several abilities that make it more flexible than shaders. Those that are relevant to the detection of edges are listed below.

OpenGL Buffer Interoperability: An instance of OpenCL, if built using an OpenGL context, will have direct access to buffers from OpenGL for both reading and writing. However, locking the buffers is necessary in some cases.
Read Freedom: With access to all buffers on the GPU, OpenCL can access multiple buffers from within the same kernel. OpenCL can also request data from buffers out of order, unlike shaders.

Write Freedom: As long as there is enough space, a single OpenCL kernel can output any number of values to any number of buffers.

With these abilities, an OpenCL-based edge detection algorithm can make use of data already on the GPU, removing the need for data duplication. The calculations also need not be repeated, since multiple values can be output from a single kernel.

Edge Data Structures

Because almost all the data needed for the edge detection step already exists on the GPU for normal mesh rendering, OpenCL's edge mesh structure need only contain the vertex indices for each edge's adjacency information.

i0: index of first vertex in the edge
i1: index of second vertex in the edge
i2: index of final vertex that, with i0 and i1, indexes the first polygon
i3: index of final vertex that, with i0 and i1, indexes the second polygon

These values can be used to index directly into the vertex and normal buffers. We only need one copy of them because OpenCL can output multiple values from the same kernel. This can represent a significant reduction in the memory requirements of the edge detection.

Because OpenCL cannot export vertices to the rendering pipeline directly, the output values, both edge quads and degenerate quads must be temporarily stored to the GPU's memory for rendering in another pass. This edge out buffer must be created to hold 4E four dimensional vertices (or 16E floats) values, where E is the number of edges in the mesh.

Edge Detection and Rendering

The edge detection tests are exactly the same as those in McGuire and Hughes' method with one exception. Since index values are available, the boundary and marked edge tests can be accomplished by checking for index equality of i0 and i3, rather than checking the indexed vertices.

As before, if the edge is not drawable, a degenerate quad should be exported to the edge out buffer made up of the vertex (0, 0, -1, 1). For drawable edges, they should be expanded as in McGuire and Hughes' method before being exported. In both cases, all the vertices in the projection-space quads are exported in a single step to their appropriate location in the edge out buffer, using the same identifying index used to get the input edge information.

The edge detection kernel will need access to the ModelViewProjection matrix and the viewport dimensions to calculate the screen-space position of the vertices. Since OpenCL is not part of the shading environment, these values must be manually sent to the kernel each frame. The edge out buffer, after the edge detection step is complete, will contain sets of four projection-space vertices representing the edge quads or degenerate quads. Therefore, when rendering the edge out buffer as quads, the vertex shader should apply no transforms on the vertices at all.

McGuire and Hughes-based Capping

Beyond edges, the McGuire and Hughes' capping method can also be implemented in OpenCL. In fact, it requires no additional data to integrate it into the edge detection system described above. It only requires two additional output buffers capable of storing at least 3E four dimensional vertices (12E floats) each, where E is the number of edges in the mesh. Again, this is due to OpenCL's inability to export data directly to the graphics pipeline.

If the edge was detected drawable, the two edge half-caps can be immediately generated. The normal information needed can be accessed using the index values i0 and i1 on the normal buffer. From then on, the vertex creation process is identical. As with the edges, the rendering passes for the cap buffers can simply pass their vertex data through the vertex shader.

Edge Adjacency Capping

OpenCL's abilities to access and manipulate many buffers provides for another method of edge capping. In McGuire and Hughes' caps, if the projected normal did not correspond well to the curvature of the mesh, caps could be generated on the wrong side of the edge. With access all edges, we can make a list of connected edges. Using this, the capping process is no longer a blind activity that generates two half-caps per drawable edge. Instead, caps are created relative to the vertex to which they connect. With both edges connecting at that vertex available, we can determine exactly which caps should be created, and exactly what size to make them so they always fill the gap perfectly. This eliminates the need for vertex normals and prevents the problems with caps being generated incorrectly. A number of edge orientations and the caps they form are displayed in figure 5.

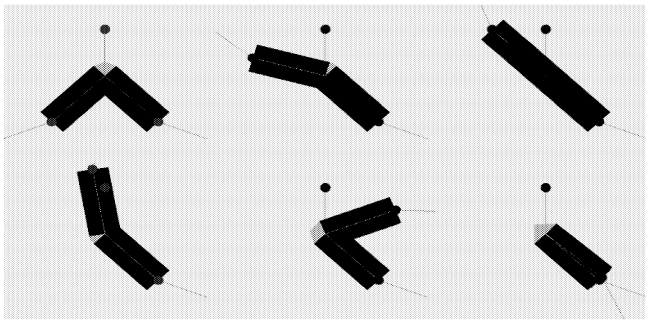


Figure 5: No matter the orientation of the two edges or the normal, a perfect cap is formed to fill the gap every time.

A cap buffer must be preprocessed. It stores indices like the edge buffer. However, the indices reference index sets in the edge buffer. This makes a single potential cap from every pair of connected edges. Each vertex in this "cap mesh" contains:

e0: index of first edge
e1: index of second edge

As with the edge buffer, the cap buffer needs a corresponding cap out buffer to store the output caps for later rendering. It must be big enough to hold 4C four dimensional vertices (16C floats), where C is the number of cap vertices in the cap buffer.

Cap Detection

In this method of capping, only those caps for whom both connected edges are drawable should be drawn. Thankfully, because the edge detection step must store its output data temporarily, it is trivial to determine if both are drawable. After obtaining the projection-space edge quads from both connected edges, drawability can be determined by checking one of the vertices in each for equality with the degenerate vertex (0, 0, -1, 1). To reduce the test to a single comparison, during the edge test, degenerate quads can be exported that use the vertex (NaN, 0, -1, 1). Then drawability can be determined by checking the x component of the first vertex of each quad for being NaN (Not a Number). The values remain degenerate in this variation, so they will not generate artifacts when rendering the edges.

Cap Creation and Rendering

If both edges are drawable, the cap should be created. The projection-space edge quads are used to determine where to place the cap. However, there is no guarantee of the edges' orientations. We must ensure they are aligned with each other, which for this paper means that the i1 side of edge one is connected to the i0 side of edge two. This is easily done by using the cap buffer's indices to look up the edge indices for each edge and determining if one or both edges' output values should be flipped before use.

Once aligned, the projection-space edge points can be transformed to screen-space using homogeneous division and multiplication by the viewport width and height. At this point, the values can differ significantly depending on the usage of half-quads. If half-quads were used in addition to full quads, a large amount of logic or additional data is needed to successfully determine the exact connection point for the aligned edges. For the sake of time, we assume that only full quads were used.

For the first edge quad, the point halfway between the first and last screen-space edge points represents the left edge point, called sL. The halfway point between the second and third screen-space edge points for the first edge quad is the point of connection for the two edges, called sM. Finally, the halfway point between the second and third screen-space edge points of the second edge quad represent the right edge point, called sR.

The "middle vector" can be determined from sL, sM, and sR. It takes the place of the screen-space normal in the cap creation process. It is the vector bisecting the two edge vectors, inverted. It is calculated via the formula below.

```
vec2 middleVector = -normalize(normalize(sL.xy - sM.xy) +
    normalize(sR.xy - sM.xy));
```

Once the middle vector is obtained, the only remaining required values are the two perpendicular vectors, one for each edge. After calculation, they have to be

aligned to point in the same direction as the middle vector. They are calculated as below.

```
vec2 p0 = normalize((vec2)(sL.y - sM.y, sM.x - sL.x));
p0 = p0 * sign(dot(p0, middleVector));

vec2 p1 = normalize((vec2)(sR.y - sM.y, sM.x - sR.x));
p1 = p1 * sign(dot(p1, middleVector));
```

With all these values available, the four cap vertices can be created, as listed in the table below. These are all exported to the cap out buffer in one pass. As with the edges, those caps that should not be drawn should output a degenerate quad using the vertex (0, 0, -1, 1).

```
0  vec4((sM.xy + middleVector) / sM.w * vec2(Width,
       Height), sM.zw)
1  vec4((sM.xy + p0) / sM.w * vec2(Width, Height), sM.zw)
2  vec4(sM.xy / sM.w * vec2(Width, Height), sM.zw)
3  vec4((sM.xy + p1) / sM.w * vec2(Width, Height), sM.zw)
```

Comparative Analysis

We compare the memory usage, speed, and rendered output of McGuire and Hughes' method in shaders, McGuire and Hughes' method in OpenCL, and our new OpenCL method.

Memory Usage

Calculating the total memory footprint of both edges and caps is dependent on the number of possible edges and caps, the view direction, and the type of mesh in question. We compare the number of bits required to store the data necessary both to calculate the edges and caps, and in the case of OpenCL, the bits required to temporarily store the output for later rendering. The shader implementation must have some memory to store the output of its calculations, but it is hidden by the implementation and not directly measurable. 32-bit float and integer types were assumed. Values are in bits.

McGuire & Hughes (shader)		McGuire & Hughes (OpenCL)		New OpenCL	
Edge	*Cap*	*Edge*	*Cap*	*Edge*	*Cap*
2688	96	640	512	640	576

At this point, the edge to cap ratios differ. The number of caps necessary to test is significantly higher for the new OpenCL capping method, whereas in both methods that implement McGuire and Hughes' capping, the cap to edge ratio is always 2. To get a good sample, a variety of meshes were tested, rendered from a camera position of (5, 5, 5) with the unit-sized mesh at the origin. From this, the below table illustrates the total number of bits used for each type of object in all three methods.

	McGuire & Hughes (shader)	McGuire & Hughes (OpenCL)	New OpenCL
Cube	69120	39936	29184
Cylinder	276480	159744	245760
Cone	184320	106496	381952
Quad Sphere	5806080	3354624	5289984
Ico Sphere	5529600	3194880	6741120
Teapot	3398400	1963520	3301120
Monkey	4173120	2411136	5067648
Bunny	59938560	34631168	75118720

By converting these bit requirements relative to the amount used by our base case (McGuire and Hughes' algorithm in shader), we get a list of memory usage ratios.

	M&H (OpenCL) to M&H (Shader)	New OpenCL to M&H (Shader)
Cube	0.577	0.422
Cylinder	0.577	0.888
Cone	0.577	2.07
Quad Sphere	0.577	0.911
Ico Sphere	0.577	1.21
Teapot	0.577	0.971
Monkey	0.577	1.21
Bunny	0.577	1.25

Implementing McGuire and Hughes' algorithm on the GPU, even with the extra temporary storage requirements, yields a significant memory savings. The new OpenCL capping method, on the other hand, varies significantly based on the number of connections per point in the mesh. Overall, the memory usage is about the same.

Speed

To measure the speed of the three methods, we compare the framerates from the same view direction for the same set of example objects.

	McGuire & Hughes (shader)	McGuire & Hughes (OpenCL)	New OpenCL
Cube	1173	762	760
Cylinder	1103	735	733
Cone	1244	670	724
Quad Sphere	713	584	416
Ico Sphere	739	593	360
Teapot	850	631	464
Monkey	770	634	334
Bunny	134	159	31

Clearly, the shader implementation is the fastest for most meshes. Only for very complicated meshes is the McGuire and Hughes' method implemented in OpenCL faster than the shader. All tests of the new OpenCL capping method resulted in significantly slower speeds than the shader-based edge detection.

The reasons for the speed differences are numerous. OpenCL cannot short-circuit code, so the worst-case calculation path is effectively always used. The new OpenCL capping method relied upon short-circuiting for its theoretically faster speeds. McGuire and Hughes' algorithm did not rely on short-circuiting, and so had effectively fewer instructions than the new OpenCL method. Additionally, memory access in the shader is always cached, because it is accessed in order. Both OpenCL methods access the memory out of order to save on memory usage. This prevents the manual caching methods from being possible. Other issues, such as render deferment and the newness of the OpenCL API likely contributed to the slower results.

Rendered Output

McGuire and Hughes' algorithm, whether implemented via shader or OpenCL, generate identical output. In this section, we compare their quantity of rendered items and the quality of the output with the new OpenCL capping method.

Rendered Quantity

The number of rendered objects effects the final rendering speed to some degree. All three methods generate the same number of edge quads from a given angle. However, though the new OpenCL capping method has about double the number of caps to check, the number of output caps is usually about half that of McGuire and Hughes' capping method.

Rendered Quality

Figure 6 shows a sample object rendered with full quads. The new OpenCL method on the left shows all edge gaps completely filled. The other artifacts are from edge quads pushing through the mesh. On the right, McGuire and Hughes' method of capping is shown. Notice the missing caps on the eyebrows and extraneous caps sticking off the ends of other edges.

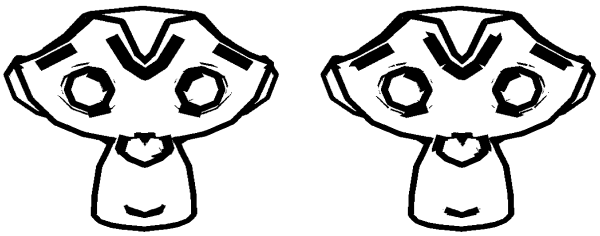


Figure 6: On the left, the new OpenCL capping method has far fewer artifacts than the McGuire and Hughes' capping method displayed on the right.

The quality can be further enhanced with the use of half-quads on contour edges, but this would result in a further slowdown due to the complexity of calculating the correct caps when there is such variability in the shape of connecting edges.

Conclusion

Our new method of capping is more accurate, but much slower. It also has no real memory advantages over the shader implementation. However, using OpenCL to implement McGuire and Hughes' algorithm holds much promise. It saves a lot of memory and can render faster than shaders for some very complicated meshes. Future improvements to OpenCL and the algorithm may result in even higher speeds and better performance for simpler meshes.

Future Work

OpenCL was used in this paper to take the place of data texture lookups and geometry shader usage suggested by the future work of McGuire and Hughes' paper. Further, texture memory access is cached on GPUs, unlike other buffers. The new OpenCL capping method could be implemented via geometry shaders and data textures, which provides the higher accuracy of the new capping method with speed closer to those of the shader-based algorithm.

If one implemented McGuire and Hughes' algorithm on the GPU such that the input data was duplicated rather than indexed, OpenCL's manual caching methods could be used. This should result in a much higher speed, but at the cost of higher memory usage.

Microsoft's DirectCompute shaders have many of the same abilities as OpenCL and any of the discussed methods could be implemented using that technology. One advantage it has over OpenCL is the ability to export data directly to the graphics pipeline. This will allow the edge and cap rendering passes to be integrated into the computation passes. That ability to skip two render passes could increase the speed slightly.

Chris Peters [Peters 10] suggested a capping method that preprocessed edges so that both ends contain a "next" index. The next index points to an adjacent edge connected to the same end point on a triangle. After doing the edge compute pass, each drawable edge would be operated on twice, once for each end. At each end of each drawable edge, the next index would be used to check the next edge for its drawability. If it is drawable, a cap is formed between those two edges. If it is not drawable, the next index of the second edge is checked, and so on, until either a drawable edge is found, or the original edge is reached. If a drawable edge connects to no drawable edges, no cap is drawn. This system should give the same accuracy caps as new capping method, but without needing to store or check every possible combination. The worst case number of memory accesses is C + 1 + DATA, where C is the number of connecting edges and DATA is the other drawable edge's data needed to form the cap. This technique would still use uncached GPU memory, but requires vastly fewer memory accesses for the capping operation. This method could also potentially lead to faster speeds for more complex meshes.

References

[Gooch et al. 99] - Gooch, Bruce, Peter-Pike J. Sloan, Amy Gooch, Peter Shirley, and Richard Riesenfeld. 1999. "Interactive Technical Illustration." http://www.ppsloan.org/publications/iti99.pdf.

[Hall 03] - Hall, Tom. 2003. "Silhouette Tracking." http://www.bytegeistsoftware.com/various/SilhouetteTracking.pdf.

[Hertzmann and Zorin 00] - Hertzmann, Aaron, and Denis Zorin. 2000. "Illustrating smooth surfaces." http://mrl.nyu.edu/~dzorin/papers/hertzmann2000iss.pdf.

[Lander 01] - Lander, Jeff. 2001. "Images from deep in the programmer's cave." Game Developer. May. 23-28.

[Markosian et al. 97] - Markosian, Lee, Michael A. Kowalski, Samuel J. Trychin, Lubomir D. Bourdev, Daniel Goldstein, and John F. Hughes. 1997. "Real-Time Nonphotorealistic Rendering." ftp://ftp.cs.brown.edu/pub/papers/graphics/research/sig97-npr.pdf.

[McGuire and Hughes 04] - McGuire, Morgan and John F. Hughes. 2004. "Hardware-Determined Feature Edges." http://graphics.cs.williams.edu/papers/EdgesNPAR04/edges-NPAR04.pdf.

[Peters 10] - Peters, Chris. 2010. Personal Communication.

[Sander et al. 00] - Sander, Pedro V., Xianfeng Gu, Steven J. Gortler, Hugues Hoppe, and John Snyder. 2000. "Silhouette Clipping." http://research.microsoft.com/en-us/um/people/hoppe/silclip.pdf.

Appendix: Depth-Scaled Edge Thickness

When scaling an edge's thickness in screen-space, two types of artifacts occur. First, perspective scaling is lost on objects, possibly confusing the user about the distance of the object. Second, as a mesh gets further from the camera, the edges will take up such a large portion of the displayed area that they overpower the mesh itself, as seen in Figure 7.

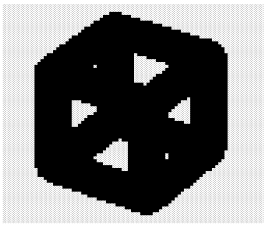


Figure 7: This cube is far away from the camera, yet the edges remain the same pixel width, overpowering the mesh.

To prevent this, we introduce a depth-based scaling factor into the width calculation of the edges and caps. This value would be multiplied into the perpendicular vector, screen-space normal vectors, and the middle vector at the time of output.

In the process of creating the values necessary to create the edge, the edge points v0 and v1 are taken from model-space, through view-space, and into projection space, before later being sent to screen-space. Since the projection-space location is known, we can get the view-space depth from the projection-space z coordinate multiplied through the inverse projection matrix. As it turns out, the inverse projection matrix as applied to the z coordinate is the projection-space w coordinate multiplied by -1. To transform this value into a proper scaling factor, it must be inverted and negated due to the orientation of the camera in view-space. Further, we can prevent the creation of edges thinner than a single pixel by capping the minimum value at 1. The resulting scaling factor is displayed below:

```
depthScalingFactor = max(1.0, 1.0 / projected.w);
```

A different scaling factor is needed for both ends of the edge. In the equation above, projected represents the s0 or s1 values before they have been sent to screen-space.

The user can then freely change the value over the projected point's w to whatever custom thickness scaling factor they desire. Edges and caps will realistically grow smaller as they grow more distant, but never fully disappear.

Enhanced Cellular Automata for Image Noise Removal

Abdel latif Abu Dalhoum
a.latif@ju.edu.jo

Ibraheem Al-Dhamari
ibr_ex@yahoo.com

Department of Computer Science, King Abdulla II School for Information Technology
University of Jordan
Amman, Jordan

Alfonso Ortega
manuel.alfonseca@uam.es

Manuel Alfonseca
alfonso.ortega@uam.es

Department of Computer Science, Escuela Politécnica Superior
Universidad Autónoma de
Madrid, Spain

KEYWORDS

Noise removal, Salt and pepper, Cellular Automata, Uniform.

ABSTRACT

Cellular Automata (CA) are a type of complex systems based on simple and uniformly interconnected cells. They provide an excellent method to perform complex computations in a simple way. CA can be used in image processing, because of the simplicity of mapping a digital image to a cellular automata and the ability of applying different image processing operations in real time. Noise removal is considered to be an important application of image processing; digital images can be corrupted by different types of noise during the image acquisition or transmission. In this paper we propose a CA model that deals with two types of noise: salt and pepper noise, and uniform noise. Our results show that the proposed model removes more noise, compared with previous models.

1. INTRODUCTION

1.1 Noise in Digital images

Digital images may be corrupted by different types of noise during their acquisition or transmission. Some pixel values may be altered (become noisy pixels), while others remain unchanged. There are two common types of noise: uniform noise and salt and pepper noise.

In uniform noise, the corrupted pixel may take any value from 0 to the maximum allowed value (we are assuming a gray scaled image). In salt and pepper noise, the corrupted pixel may take just one of two different values: black or white.

In order to remove unwanted noise and enhance the image quality, the median filter has been used (Pitas et al., 1990; Astola et al., 1997; Gonzalez and Woods, 2008). The median filter is a nonlinear effective filter used in noise removal, whose main disadvantage is that it blurs fine details or destroys edges while filtering out the noise. To preserve details while noise is reduced, many researchers have proposed different ideas (Ko et al., 1991; Chen et al., 1999; Eng et al.,2000). With the median filter, the intensities of the neighboring pixels are sorted and a median value is assigned to the center pixel.

1.2 Cellular Automata (CA)

Cellular Automata (CA) are a decentralized computing model that provides an excellent platform for performing complex computations with the help of just local information. CA are made up of interconnected cells, each of which contains an automaton, a simple machine able to perform simple computations. Each automaton has a state, which changes with time based on the states of its neighboring cells (see figure 1) [9]. The CA model transition rule determines the neighborhood relationship between the automata. Each automaton changes its state (its value) at time (t) based on the state at the previous time (t-1) of its neighbor cells (see figure 2). Introduced by John Conway in 1970, the Game-of-Life (GOL) is the most widely known example of CA (Wolfram, 2002; Sarkar, 2000; Gardner, 1970) CA have many applications for a wide variety of fields.

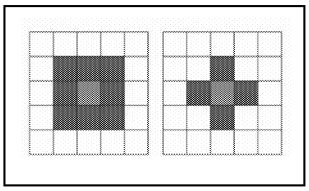


Figure 1.Common CA neighborhoods. MOORE (at the left) and Von Neumann (to the right).

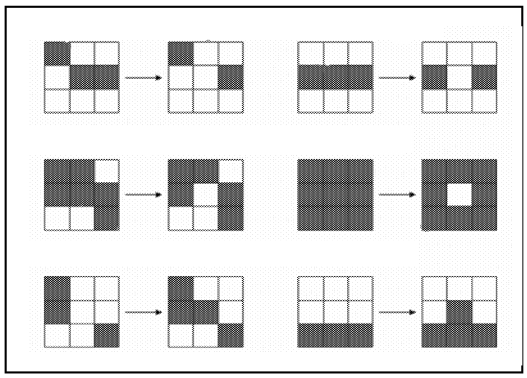


Figure 2. CA transition rule examples. The center automaton state will be:

0 if the cell has <=2 neighbors at state 1

0 if the cell has >=4 neighbors at state 1

1 if the cell has 3 neighbors at state 1

2. RELATED WORK

2.1 Uniform noise removal filter

In (Chang et al., 2008), the authors have proposed a new image-de-noising filter based on the standard median (SM) filter. In their method, a threshold and the standard median is used for noise detection and to change the original pixel value to a new value closer or similar to the standard median. Inspired by the Tri-State Median (TSM) (Chen et al. 1999), they have proved that their filter improves the SM filter, the Center Weighted Median filter (CWM) (Ko et al., 1991), and the TSM filter. In CWM, the value of the center pixel will be repeated several times. The number of times to be repeated is called the center weight. In the TSM filter:

$$TSM_{ij} = \begin{cases} X_{ij} & if \quad T \geq d_1; \\ CWM_{ij} & if \quad d_2 \leq T < d_1; \\ SM_{ij} & if \; T < d_2. \end{cases}$$

Where, $d_1 = \left| x_{ij} - SM_{ij} \right|$, $d_2 = \left| x_{ij} - CWM_{ij} \right|$ and T is a value between 0 and 255. Figure 3 shows Chang et al. proposed filter, where Xij is the center pixel value, WS is the number of the neighbors (usually 9), Ri is the ith element in the sorted neighbors sequence, rank(Xij) is the index of Xij in the sorted neighbors sequence, and the threshold value T equals 15.

Chang et al proposed filter

$$AM_{ij} = \begin{cases} \left\lfloor SM_{ij} - \left| R_{\frac{WS+1}{2}} - R_{\frac{WS-1}{2}} \right| \mathrm{X} \dfrac{rank(X_{ij}) - \frac{WS+1}{2}}{\frac{WS-1}{2}} \right\rfloor & if \; rank(X_{ij}) \leq \dfrac{WS+1}{2}; \\ \left\lfloor SM_{ij} - \left| R_{\frac{WS+1}{2}+1} - R_{\frac{WS+1}{2}} \right| \mathrm{X} \dfrac{rank(X_{ij}) - \frac{WS+1}{2}}{\frac{WS-1}{2}} \right\rfloor & if \; rank(X_{ij}) > \dfrac{WS+1}{2}. \end{cases}$$

Followed by:

$$AM_{ij} = \begin{cases} AM_{ij} & if \; \left| X_{ij} - AM_{ij} \right| \geq T; \\ X_{ij} & if \; \left| X_{ij} - AM_{ij} \right| < T. \end{cases}$$

Figure 3. Chang et al. filter

2.2 Salt and paper noise CA de-noising model

In (Liu et al, 2008), the authors proposed a novel de-noising algorithm based on CA to filter images with salt-pepper noise. Their CA local transition function is based on Moore neighborhood. They have evaluated their approach by using the hamming distance to compare it with the classical median filter, and showed that their algorithm has better de-noising effects, especially when the noise density is bigger than 40%. Figure 4 shows their transition function.

3. THE PROPOSED CA MODEL

We introduce a novel CA model for noise removal. Our proposed model deals with both types of noise; salt and pepper noise and uniform noise. We first detect the type of noise by computing the histogram of the noisy image. If the most frequent values in the image histogram are black or white, we conclude that the image contains salt and pepper noise, otherwise it contains uniform noise. The next step is removing the noise by using the CA transition rules described in figure 5.

Our proposed CA model checks the noise type and response correctly for each type. If the noise type is

uniform noise, we exclude the maximum and minimum values from the neighbors then compute the median of the remaining values, after that assign it to the current automaton state. If the noise type is salt and pepper, we check if the current state has black or white color which means it may corrupted by noise, then we compute the median of the neighbors that don't have black or white values and assign this median for the current automaton state, if all the neighbors has black and white values we take the average of them and assign the average to the current automaton state.

Liu et al CA transition rule

```
1. Check value of current cell Xi, j
   And values of its neighbor.
2. if X_{i,j} ≤ max (neighbors) and
      X_{i,j} > min(neighbors) then
3.    X_{i,j} stay the same.
4. elseif max(neighbors)= min(neighbors)or
      neighbors have only two states
     (max(neighbors),min(neighbors))
      then
5.   if min(neighbors) ≠ 0 then
6.      X_{i,j}=min(neighbors)
7.   elseif max(neighbors) ≠ 255 then
8.      X_{i,j}=max(neighbors)
9.   else  X_{i,j} stay the same
10.else
11.    m = mean(neighbors except
       Max(neighbors) and min(neighbors))
12.    if  abs(X_{i,j} -m)<threshold
13.        X_{i,j} stay the same
14.    else
15.        X_{i,j} =m
16.    end
17. end
18. end
19. end
20. end
```

Figure 4. Liu et al CA transition function

4. EXPERIMENTS AND RESULTS

We have implemented CA simulators for our proposed idea, for Liu et al. procedure, and for Chang et al. filter, using the well known MATLAB 7.6.0 software (Matlab, web reference). We have used two standard images in our experiments, namely Lena and Boats, as well as one of our own images (Jan), which has more details and edges, and hence makes a good test example. In this paper, however, for space reasons, we only show the Lena results.

We have compared our model with both models in (Chang et al., 2008) and (Liu et al, 2008). We have used the same measurements that they used, namely Mean Squared Error (MSE) and Hamming Distance (HD). It is well-known that when these measures are small, the technique is considered to be better. They are defined as:

$$MSE = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}(a_{ij} - b_{ij})^2}{m * n}; \quad HD = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}(a_{ij}^2 \oplus b_{ij}^2)}{m * n}$$

where a is the original image and b is the resulting image. Both images are the same size (m x n).

```
1. Check value of current cell X_{i,j}
     and values of its neighbor.
2. if uniform noise
3.    mx=max(neighbors);
4.    mn=min(neighbors);
5.    if X_{i,j} ==mx or X_{i,j} ==mn
  //take the median value of the
neighbors
6.        y=SM_{ij};
7.     end
8. else // salt and peppers noise
9.     if (X_{i,j} ==0)or ( X_{i,j} ==255)
10.       if there are neighbors
              that are not 0 nor 255
              X_{i,j}= The median of step 10;
11.       else
11.           X_{i,j}= mean(Neighbors);
          end
12.   end
13.end;
```

Figure 5. The proposed CA transition function

We should notice that MSE is more accurate than HD, because it computes the degree of difference between the two images, while HD gives only the number of different pixels in the two images. According to its purpose, we compare our model with (Chang et al., 2008) in terms of uniform noise and with (Liu et al, 2008) in terms of salt and pepper noise. For each image we have added different percentages of noise with ratios equal to 5%, 10%, 25%, 50%, 75%, 90% and 95% of the image size.

For each ratio we have produced two noisy images: one with "salt and pepper" noise, and one with "uniform" noise. We have run the simulation for each image for five iterations and recorded the results (the corrected image and the two error measurements, MSE and HD).

The results, illustrated in figures 6-9 and tables 1-2, show that our model is better than the previous models in terms

of the two measurements factors, namely, MSE and HD. The time complexity is constant, O(1), because of the parallelization that CA provide which is considered as a main advantage of CA in term of performance. We should also note that the best measurement is the human eye, especially when there is a clear difference between the resulting images.

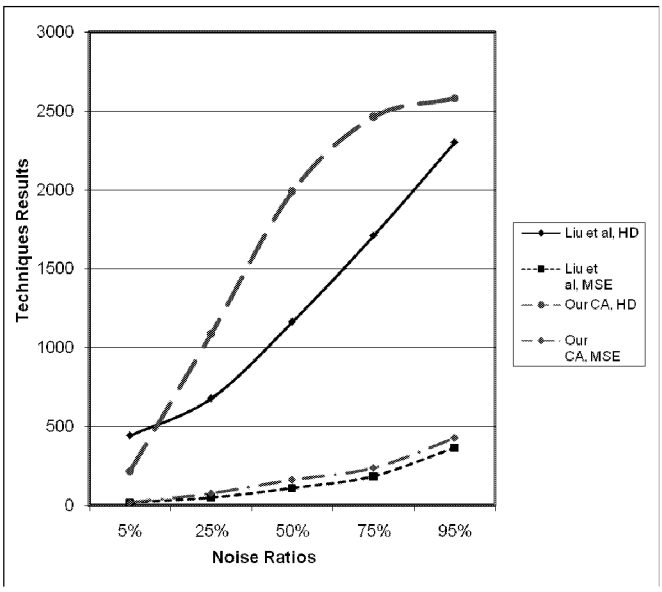


Figure 6. Comparison between our model and Liu et al. (Salt and pepper noise)

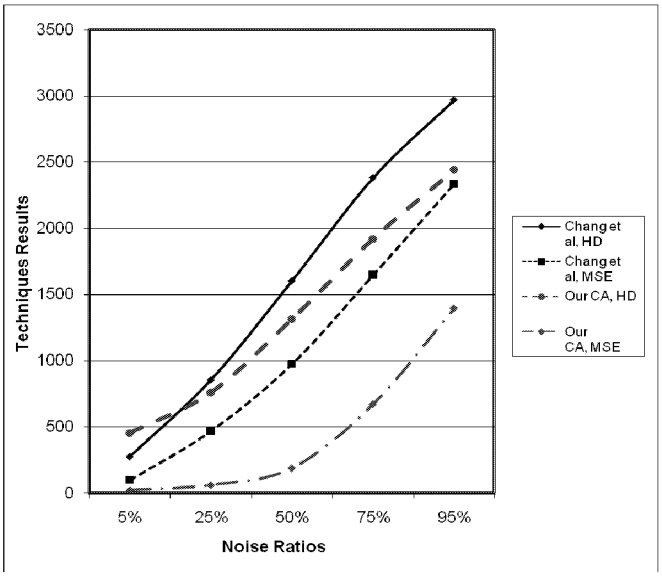


Figure 7. Comparison between our model and Chang et al. (Uniform noise)

Table 1: Sample of the results (Salt & Pepper noise)

Method	Generation 1									
	5% noise		25% noise		50% noise		75% noise		95% noise	
	HD	MSE	HD	MSE	HD	MSE	HD	MSE	HD	MSE
Liu et al	444.38	18.767	680.3	49.24	1164.35	121.9	1710	880.6	2301	5917
Our CA	220.36	21.749	1089.3	91.68	1990.25	178.8	2544	847.5	3918	5428
Method	Generation 2									
	5% noise		25% noise		50% noise		75% noise		95% noise	
	HD	MSE	HD	MSE	HD	MSE	HD	MSE	HD	MSE
Liu et al	802	26.401	1042.3	57.61	1488.4	109.9	1949	196.4	2302	2847
Our CA	551.42	20.505	1341.8	79.95	1989.48	163.8	2462	246.2	3112	2597
Method	Generation 3									
	5% noise		25% noise		50% noise		75% noise		95% noise	
	HD	MSE	HD	MSE	HD	MSE	HD	MSE	HD	MSE
Liu et al	1073.4	30.647	1296.1	62.44	1662.21	113.2	2019	185.8	2446	1114
Our CA	878.44	25.071	1636.4	78.34	1989.48	163.8	2461	239	2601	1038
Method	Generation 4									
	5% noise		25% noise		50% noise		75% noise		95% noise	
	HD	MSE	HD	MSE	HD	MSE	HD	MSE	HD	MSE
Liu et al	1268	34.143	1467.5	65.4	1662.21	115.2	2096	186.4	2446	503.1
Our CA	1103.8	27.784	1842.7	79.31	1989.48	163.8	2461	239	2601	522
Method	Generation 5									
	5% noise		25% noise		50% noise		75% noise		95% noise	
	HD	MSE	HD	MSE	HD	MSE	HD	MSE	HD	MSE
Liu et al	1415.3	37.08	1591.9	68.5	1880.14	117	2161	187.2	2495	366.7
Our CA	1245.5	30.253	1966	81.82	1989.48	163.8	2461	239	2579	427.9

Table 2: Sample of the results (Uniform noise)

Method	Generation 1									
	5% noise		25% noise		50% noise		75% noise		95% noise	
	HD	MSE	HD	MSE	HD	MSE	HD	MSE	HD	MSE
Chang et al	275.29	98.056	854.87	477.7	1602.47	1050	2381	1838	2968	2597
Our CA	452.22	27.667	756.52	277.4	1312.68	898.9	1920	1767	2440	2563
Method	Generation 2									
	5% noise		25% noise		50% noise		75% noise		95% noise	
	HD	MSE	HD	MSE	HD	MSE	HD	MSE	HD	MSE
Chang et al	300.51	99.151	901.64	469	1697.08	983.2	2559	1692	3245	2430
Our CA	785.96	21.653	1002	102.5	1457.19	463.5	2017	1184	2527	1962
Method	Generation 3									
	5% noise		25% noise		50% noise		75% noise		95% noise	
	HD	MSE	HD	MSE	HD	MSE	HD	MSE	HD	MSE
Chang et al	311.23	99.898	925.37	470.5	1746.37	974.7	2672	1660	3398	2374
Our CA	1017.6	21.653	1239.3	67.12	1635.57	289.3	2131	897.5	2606	1652
Method	Generation 4									
	5% noise		25% noise		50% noise		75% noise		95% noise	
	HD	MSE	HD	MSE	HD	MSE	HD	MSE	HD	MSE
Chang et al	315.34	99.896	937.2	471.9	1774.83	975.6	2738	1650	3484	2348
Our CA	1164.3	23.681	1397.2	60.94	1773.6	218.5	2232	752.2	2675	1488
Method	Generation 5									
	5% noise		25% noise		50% noise		75% noise		95% noise	
	HD	MSE	HD	MSE	HD	MSE	HD	MSE	HD	MSE
Chang et al	316.95	100.31	941.93	472.7	1789.56	977.4	2777	1646	3537	2334
Our CA	1262.4	26.129	1494.3	61.06	1863.93	188.1	2300	670.9	2727	1394

5. CONCLUSION AND FUTURE WORK

In this paper we have introduced a novel CA model for image noise removal. Our model deals successfully with both "salt and pepper" noise and "uniform" noise. We have shown that our model is better than Chang model, and almost as good as Liu model. Our CA model has successfully removed the two types of noise in different ratios. As a future work, we will enhance the current proposed model in terms of performance and accuracy, besides generalizing it to deal with more noise types.

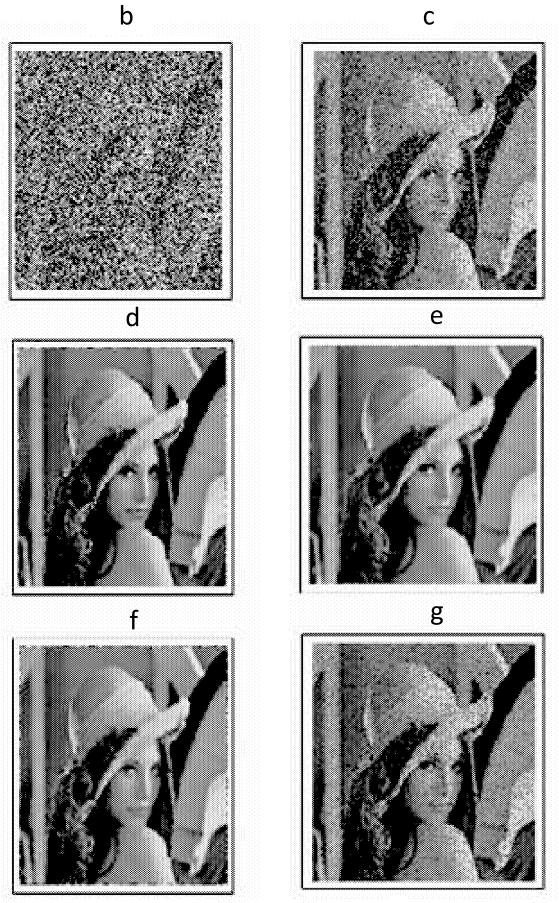


Figure 8. Another Sample of the results.
a. Original Lena image.
b. Lena image with 75% of salt and pepper noise.
c. Lena image with 25% of uniform noise.
d. Result of applying our CA model on b.
e. Result of applying our CA model on c.
f. Result of applying Liu et al filter on b.
g. Result of applying Chang et al filter on c.

REFERENCES

Astola J. and Kuosmanen P., 1997, "Fundamentals of Nonlinear Digital Filtering", Boca Raton, FL: CRC..

Chang C., Hsiao J. and Hsieh C., 2008, "An Adaptive Median Filter for Image Denoising", Second International Symposium on Intelligent Information Technology Application, IEEE, DOI 10.1109/IITA.2008.259.

Chen T, Ma K. and Chen L., "Tri-state median filter for image denoising", IEEE Transactions on Image Processing, Vol. 8, No. 12, Dec. 1999, pp. 1834-1838.

Eng H. and Ma K., 2000, "Noise adaptive soft-switching median filter for image denoising", IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 4, 2000, pp. 2175-2178.

Gardner M., 1970, "The Fantastic Combinations of John Conway's New Solitaire Game Life", Scientific American, vol. 223, issue 4, pp. 120-123.

Gonzalez R. and Woods R., 2008, "Digital Image Processing", 3rd edition.

Ko S. and Lee Y., 1991, "Center weighted median filters and their applications to image enhancement", IEEE Trans. Circuits Syst., vol. 38, pp.: 984–993, 1991.

Pitas I. and Venetsanopoulos A., 1990, "Nonlinear Digital Filters: Principles and Applications" . Boston, MA: Kluwer.

Sarkar S., 2000, "a brief history of cellular automata", CSUR vol. 32, issue.1, pp. 80 - 107.

Songtao L., Chen H. and Yang S., 2008, "An Effective Filtering Algorithm for Image Salt-pepper Noises Based on Cellular Automata", 2008 Congress on Image and Signal Processing, IEEE, DOI 10.1109/CISP.2008.263.

Wolfram, 2002, "a New Kind of Science", Wolfram Media.

WEB REFERENCES

Matlab web page, www.mathworks.com/products/matlab, Last access: May 30, 2010.

AUTHOR LISTING

AUTHOR LISTING