Zombs

Jordan Gasch Senior Project 2011

Team Zombs

Alan DeLonga, Evan Klein, Jordan, Gasch, Reece Engle

Introduction

Start Screen for Zombs

The *Zombs* project was initially conceived as a proposal in our CPE476++ class, of a game that would be a mix between bomber-man and Diablo II, using cell shading. As the project progressed we chose to follow a more ominous ambiance with our models, lighting, camera angle and background sounds. By having a democratic atmosphere, where all members gave input, our project became a cohesive culmination of our ideas. Our team was initially 5 members but due to the inabilities of one of the members we only had 4 for the second phase of implementation. Our members are Alan DeLonga, Evan Kleist, Jordan Gasch, and Reece Engle. As we started the project we were given a list of technologies that were mandatory to incorporate into the game in particular:

Technology	Authors	Technology	Authors
Real-time movement/update	(All)	Level Editor	Reece Engle
View frustrum culling	Evan Kliest	Smart Camera	Evan Kliest
Particle generation	Alan DeLonga Jordan Gasch	Bomb Throwing	Evan Kliest, Alan DeLonga
Spatial data structures	Evan Kliest	HUD	Alan D., Reece E., Evan K.
Per Pixel Shading	Jordan Gasch	Inventory	Alan DeLonga
Collision detection	Evan Kliest	Wall Transparency	Evan Kliest
Models	Alan DeLonga	Model Animations	Alan DeLonga
Artificial Inteligence	Jordan Gasch	Shadows	Jordan Gasch
Sounds	Alan DeLonga, Evan Kliest	Bomb Particles	Jordan Gasch

Along with these technologies we had to create a cohesive gaming experience. Since each

member incorporated different aspects I will only go into detail on the parts I helped integrate. The game was coded the project using C++and used the libraries; SDL, libSDL_mixer (for sounds), libSDL_ttf (for text), libfreetype, libGLEW(for models, and shading). Our game was inspired by Diablo II's camera view and model interaction. We then altered it by locking the camera behind the player, dropping the angle, and pulling the camera in (instead of above) when the camera collides with walls.

With the recent boom in the video game industry, the design and development of games has become an extremely competitive and sought-after job market for software developers. In 2009 the video game industry generated over 19.6 billion dollars, this overwhelming revenue surpassed both music and movies. With this large market hardware is constantly being optimized to handle the growing needs of 'cutting-edge' video game systems.

A modern computer is capable of performing billions of computations per second. However, video games are made to represent entire worlds and sometimes can be expected to represent large sets of data that goes around. Performing operations without concern for efficiency can easily bog down a computer's processor and create an unplayable game. This makes it more important to optimize solutions than in typical programming applications.

The *Zombs* project was an appealing choice for our group with the present popularity and demand for video games, and the opportunity to put to practice many essential elements of Computer Science such as design, teamwork, and implementation. Zombies have been increasingly appearing in all forms of popular culture. Regularly depicted in horror and fantasy based entertainment, zombies have captured the interest of millions of people world wide.

Diablo II Blizzard 2000

Software projects have grown in significantly in size over the past decades, and as a result applications are rarely worked on individually in practice. This makes being productive in groups a critical skill exercised regularly while working in industry as a software developer. Our ideas were inspired by popular video games (in particular

Diablo II) and other forms entertainment as a reference for the look and feel we wanted for our game. The overwhelming workload made task prioritization and assignment an essential dynamic that continually evolved over the 20 week implementation to suit the needs of each particular milestone.

The zombie genre has become extremely popular in the current generation and this theme initially made our group very excited to begin work. However, incorporating the Zombs story-line into the game mechanics turned out to be more of a challenge than any of us had anticipated. The main issue was that each team member had a unique idea for what the game should be and these discrepancies led to a slower democratic design process.

Our main method of introducing story into the game was by adding objectives to our levels. In addition objectives also helped give the player a sense of direction. We also were able to introduce key game play concepts like how to run, fight, and navigate through the levels. Although, during play testing we did notice that our objective based system was ignored by some users. Ignoring objectives resulted in confusion and eventually detracted from the player's interest in the experience. Yet, user's that played through the levels by following the objectives, as intended, yielded more favorable feedback about the game play.

Game Play

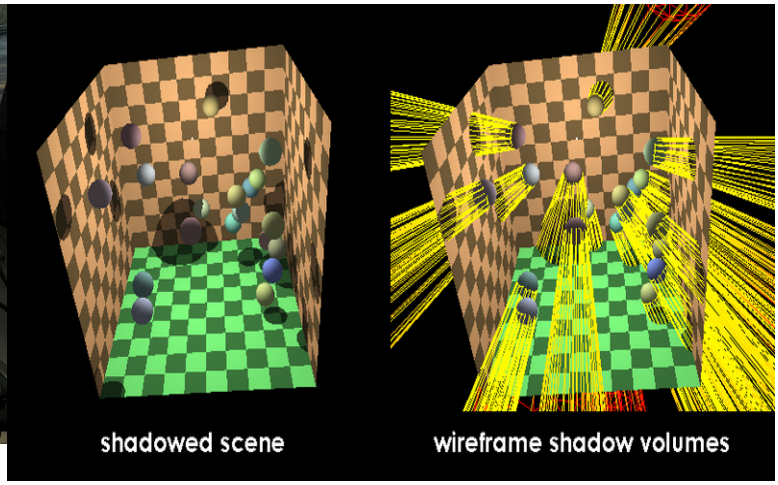
The story of our game is that the player is a survivor in a zombie apocalypse. The player wakes up in a hospital after being in an accident, the player's main objective is to find his wife who was also in the car. The player's walking is labored so he can only sprint for short distances. As the player progresses through the level the story unfolds and the player finds himself in a zombie infested hospital. Throughout the game the player view the inner thoughts of his character which help lay out this story line and give hints to game play aspects. The key game controls are "wasd" to move, mouse for camera rotation, left click melee and space bar weapon use

Since our team is creating a zombie oriented game in the horror genre the aesthetics were a very important component of our game. This includes choosing the right models, textures, and atmospheric lighting in our game. We wanted our game to give the player the feeling that they were trapped searching through an "abandoned" hospital.

Shadows

In order to create this atmosphere shadows needed to be implemented in a way which looked realistic and natural. In order to produce this effect our team originally worked with shadow volumes. This was previously implemented in Doom 3 using Carmack's stencil shadowing algorithm.

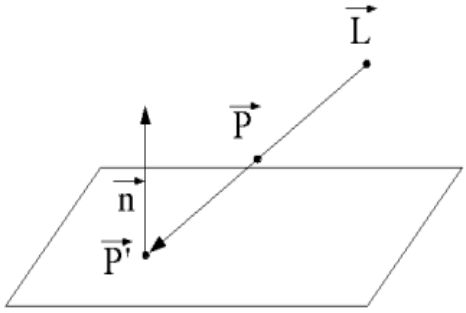
Example of Carmak's stencil shadow Doom 3



Visual representation of shadow volumes

We created a system for shadow volumes which is the process of using the stencil buffer to render shadows quickly enough to be used real time. For this technique for each light source it is necessary to make multiple passes for each light which fill the stencil buffer with values depending on how far into shadow an object is. Lit objects have a value of 0, unlit objects have a value of +1 for each shadow cast upon them. This approach was created by John Carmack of id Software and used in the development of Doom 3 (http://en.wikipedia.org/wiki/Shadow_volume) However because of the way our geometrical data was stored using our md2 loader it was nearly impossible to integrate with our project which required us to take a different approach.

In the final version of our project shadows are created using plane projected shadows. This creates a shadow matrix by taking the cross product of the light position and the ground plane and multiplying it by the dot product of the ground plane. This causes geometry to be drawn from the angle of the light allowing us to cast shadows on the floor which change based on the angle of the light.



$$\vec{x} = \vec{L} + \lambda(\vec{P} - \vec{L})$$

Projected Lighting Equation

Before shadows are drawn the depth buffer is disabled so that the shadow is drawn flat on the floor. The alpha level is then set to .5 and alpha blending is enabled. The color is set to black so that the shadow draws transparent and black on the floor.

While this technique requires us to redraw all of our object geometry it does not require multiple passes which allows our game to run at higher speeds on low end computers. Creating shadows was an extremely difficult task but the end result was well worth the time. This also allowed us to use the old drop shadows we had in place as enemy health bars which was a much needed improvement to our game design.

Projected Shadows on the ground

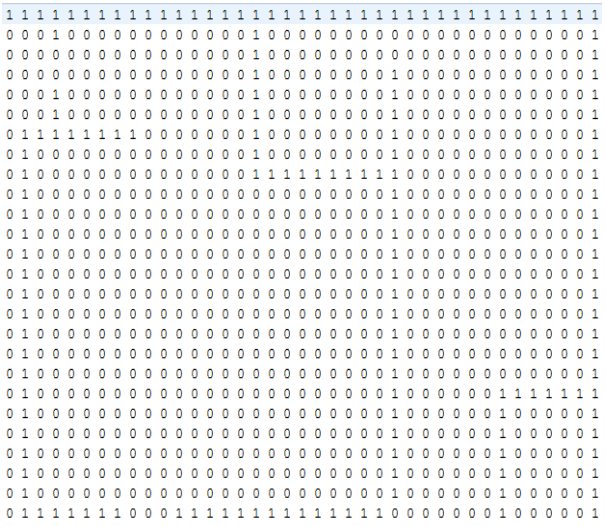
Given more time to work with the project creating a shadow volume system by rewriting our model loader would be the next step for shadows in our game. Another option would be to implement projected textures so that shadows will appear on walls. However these two approaches would have a negative impact on the speed of the game and would require the player to use a high end computer to play our game.

Artificial Intelligence

Another part of our game which was important to our team was creating artificial intelligence for the zombies. Since the enemies in this game are zombies we only needed simple AI which allows zombies to find the player from any position on the map and head in that direction. The algorithm we used was a modified A* algorithm which resembles an algorithm used Pacman. However in Pacman the ghosts are each trying to find different optimal squares where as our AI only seeks the player.

Our AI system uses an modified A* algorithm. This algorithm is modified in that instead of the zombie position finding the shortest path to the player, the player finds the shortest path to every square inside of the players awareness radius. The awareness radius is a value which changes based on the way the player plays the game. As the player is more aggressive and uses more powerful weapons the zombies become more aware of his position, increasing the awareness radius. This allows for AI computations to only be made once per frame for the player rather than per zombie. In order to account for walls we created a AI map. This map contains the locations of all the walls and objects is a file of 1's and 0's created by the level editor. This quick map allows the AI calculations to be made so that the path calculations include the zombies being unable to walk through walls and other objects.

The way this algorithm works is on every update the AI handler obtains the players position and awareness radius. This then populates the 50x50 grid with values depending on how far away the square is from the player using a breadth first approach. These values decrease with distance from the player. Zombies then check the surrounding grid and choose the space with the lowest value.



Example of AI Grid

Originally we placed doors in this map and had them disappear when they were opened however we found it much more interesting for the zombies to be waiting outside the door clawing at it if the player had high awareness. The zombies also will modify their path if blocked by other zombies. Originally zombies would stick together because they were all trying to take the same path if they were near each other. Now zombies will modify their path if blocked which allows for swarms of zombies without them being stuck together.

By breaking up our grid into a 50x50 grid it simplifies the problem of AI by allowing the zombie to do a simple check of if a space near it is open and desirable. The zombie does this by searching the 9 squares around it, choosing the square which leads to the shortest path and then moving there. This approach allows zombies to move around walls and through doors easily and find shortest path calculations per frame.

Given more time we would have liked to write the algorithm so that the zombies took into account each other and objects in the level so that they would not get stuck in doorways however this was a major undertaking and with our team being only 4 members we decided it was not the most important issue that needed to be dealt with. Another improvement that can be made is using a larger grid for the same amount of space and using objects bounding boxes to create the grid. By using a larger grid and bounding boxes of objects there would be greater accuracy in the zombies ability to find the shortest path and objects in the level would be included in finding the shortest path. Currently only walls are shown in the 50x50 grid.

Per Pixel Lighting

Lighting was another point of development that the look and feel of our game was reliant upon. We were inspired by Diablo II's lighting and the mentality that we didn't want the player to be able to see everything in our map easily however this type of a system was very difficult to create in 3D space. To solve this problem our team uses per pixel lighting calculations and GLSL to create a dark atmosphere for the player to move in.

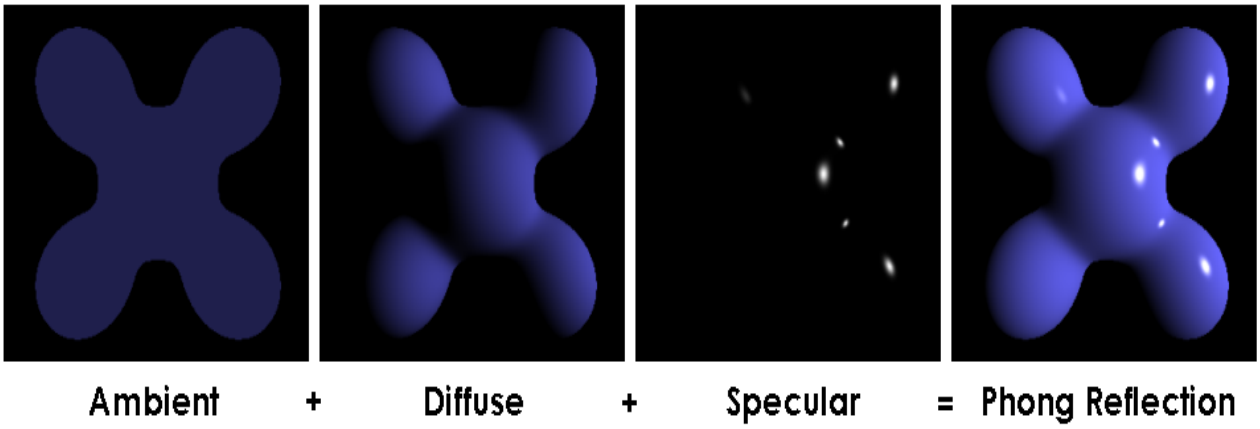
Lighting was created using GLSL per pixel calculations using the vertex and fragment shaders. Originally calculations were done using the Opengl fixed function graphics pipeline however this caused many undesired effects such as our whole level being too bright and shading being calculated per vertex rather than per pixel. This caused lights on large surfaces such as walls to be stretched and for ambient lighting to be calculated throughout the level with no attenuation.

Originally our we wanted our game to have a look and feel inspired by the comic book the walking dead. We wrote a shader which used cell shading. This effect was created by calculating lighting by clamping the Lambert term to a value of .3, .6, or 1.0 depending on which value it was closest to. While this was a cool effect it ruined the atmospheric feel of being in a dark hospital. As soon as we placed realistic lighting in our game our team knew that this was the new direction

Cell Shaded Zombs

we wanted our game to go.

The lighting was created by using per pixel lighting calculations using the Phong model for lighting including attenuation. This allowed our game to have lighting which looked realistic and dark at the same time. We struggled for a long time as a team on how light or dark our game should be and after many iterations we found values we were happy with. Working with shaders and writing our own lighting calculations was extremely satisfying and allowed us to manipulate the look and feel of the game greatly.



Visual Representation of the Phong Lighting Equation

Another problem our team ran into was the 8 light Opengl limitation. Opengl limits the number of lights in a scene to 8 which was less lights than our team wanted. This required us to write a light handling system which turns lights on and off based on their distance from the camera. This allowed our team to go past the 8 light limitation and create as many lights in our scene as we wanted so long as less than 8 lights at a time were visible from the current view point.

Zombs Multiple lighting

This gave Zombs a great deal of personality and allowed for us to easily manipulate the lighting in our scenes. Given more time to work on this project more effects and shaders

would be added to allow greater manipulation of the pixel lighting and perhaps make a more sophisticated shader to attempt cell-shading again.

Works Cited

"Diablo 2." *Blizzard Entertainment*. ttp://us.blizzard.com/en-us/games/d2/>.

Garrein, Kris. "DevMaster.net - Real-time Shadowing Techniques." *DevMaster.net - Your Source for Game Development*. DevMaster. Web. 31 May 2011. <http://www.devmaster.net/articles/shadows/>.

"Real Time Stencil Shadows with Multiple Lights." . <http://www.angelfire.com/games5/duktroa/RealTimeShadowTutorial.htm>.

Sanglard, Fabien. "ShadowMapping with GLUT and GLSL." *Fabien Sanglard's Non-blog*. <http://www.fabiensanglard.net/shadowmapping/index.php>.

"Shadows, Reflections, Lighting, Textures. Easy with OpenGL!" *OpenGL - The Industry Standard for High Performance Graphics*. <http://www.opengl.org/resources/code/samples/mjktips/TexShadowReflectLight.html>.

Staff, Inquirer. "Creative Gives Background to Doom Iii Shadow Story- The Inquirer." *THE INQUIRER - News, Reviews and Opinion for Tech Buffs*. <http://www.theinquirer.net/inquirer/news/1019517/creative-background-doom-iii-shadow-story>.

Tsiombikas, John. "Volume Shadows Tutorial." *Http://nuclear.mutantstargoat.com/articles/volume_shadows_tutorial_nuclear.pdf*. Web. <http://nuclear.mutantstargoat.com/articles/volume_shadows_tutorial_nuclear.pdf>.