LUME

Senior Project Write-Up

By Jeffrey Good

Introduction

Video game design is a constantly expanding industry, grossing more than 31.6 billion dollars annually. Game design is not only the work of large entertainment companies but has also become available to small companies, freelancing, and individuals. This growing industry is one of the major contributors to new and upcoming technologies in the computer science field. Computer games and graphics alike are constantly evolving with the invention of better hardware and more efficient algorithms. The challenge in game design is to create something unique and up-to-date with the latest technologies. Adventure games, in particular, require a large amount of content in order to drive the objective-based system. It is this genre of video game that provides a large amount of revenue for the game industry, often spawning from story-lines of books and movies. Because of the impact of this genre and the expansiveness of the game industry as a whole, our team chose to implement a 3D interactive adventure game for our senior project.

Project Lume was a two quarter long real-time graphics project. Our design team consisted of seven team members: Mike Buerli, Brent Dimapilis, Trent Ellingsen, Jeff Good, Teal Owyang, Jonathan Rawson, and Ryan Schroeder. As a team, we successfully developed a fully playable computer game, along with the necessary engine, content, and tools needed to create such a game. This project not only produced a unique and interactive 3d game, but also gave team members invaluable experience with computer gaming and graphics technologies.

Project Overview

Lume is a unique 3D adventure game created in C++ and OpenGL. The game exhibits a large array of computer graphics technologies complemented by a strong story and distinctive aesthetic appeal. Lume also uses a different style of controls by implementing both first and third person perspectives, while still keeping the controls intuitive and fluid.

Lume is centered around a creation and sandbox feel intended to give the user a fully immersing adventure experience. Players are encouraged to build upon the world in order to reach new objectives, checkpoints, and gain more abilities. The world originally starts dark and desolate, however, when the user interacts with buildings and moves throughout the world they give light and life back to the world providing the player a strong sense of influence over their surroundings. Throughout the course of the game, the player gains more abilities through leveling up including: the ability to control of moving platforms, further extend blocks, and jump through blocks that have been built. All of these abilities must be utilized in order for the player to reach new heights and complete the various objectives for each level.

The main objective of Lume is to reach the top of a collection of buildings which comprise a level to harness more energy. Focusing on this simple idea is imperative in game development. Abilities and controls must be simple and engaging; a game must begin with a simple concept and build upon that concept with features that help bring more value to the game by either improving basic game play or strengthening the story. The focus in Lume was the ability to extend blocks from any building in the world and climb up them, allowing the user to choose their own path through various levels. Throughout the development of Lume, other features were added to help strengthen the story or improve the game play.

Look and feel is another vital component to game development. A game must have a very cohesive theme and style in order to keep the user engaged. Lume's theme focuses on futuristic and abstract lighting with heavy contrasts between light and dark. Objects are constantly pulsating with a darker shade of light, which contrasts the bright glow of the edges surrounding buildings. The color pallet of the world possesses a variety of neon colors which help convey a futuristic style. The look and feel of a game must also help convey the story. Lume focuses on tying a relationship between the dark and bland qualities of the world to objects that KOG is controlling. As the player progresses through the game they harness energy from KOG and create a brighter and more colorful world with the newly harnessed energy providing a stark contrast between light and dark. Players can add color to the world simply by building blocks on any building. Organic, colorful patterns extend from each block a player creates in the world. Lastly, to further convey the differences between KOG and the player, objects associated with KOG are much more linear and rigid, while objects associated with the player are more organic and abstract.

The last element and platform for which a game is built upon is the storyline. The KOG story was developed by teammates throughout the two quarters plotting out much more detail than is visible in the game. To be brief, the story is of a future civilization, which is under the complete control of KOG, who's only hope of survival is Lume (the player).

The story takes on the classic battle of humanity versus technology (not evident until the end of the game) as well as nature versus industry. The story starts out in the year 4200, where KOG, a privatized company now controls all civilization. All of earths resources have been depleted and the world is now solely based on energy. The intro scene for Lume starts with an image of a door labeled Project Lume. Project Lume is an experiment conducted by KOG to create a new and more efficient way of storing energy. Lume (the character) is the first prototype of this experiment which KOG quickly realizes is a failure due to Lume's unexpected ability to manipulate energy. A computer terminal is seen activating a program called Insight. Insight is a computer AI designed back in 3119 at the creation of KOG, designed as a fail-safe to disable KOG should it ever gain too much control. Presently, one-thousand years after KOG's rise to power, Insight finally sees its chance to take KOG down using Project Lume. The intro shows Lume falling from the sky (down from the upper KOG city) and then awakening in the tutorial level where the game begins. The first objective of the game is to download Insight (initially the player does not know what they are downloading). Once Insight is downloaded it talks to the player via the Heads Up Display (HUD), giving the player feedback and an introduction to the game's plot. Insight guides you through a series of levels, in which you harness an increasing amount of KOG's energy. Once all the energy is collected you can reach the upper city, where the player is given the opportunity to finally defeat KOG. After completing the game, it is revealed that KOG stands for Komputer Organized Government and that KOG is a Komputer (next gen computer). By shutting KOG down, all energy is relinquished back to the earth, allowing humanity and nature to start once again.

Related Works

The general look and feel for Lume was inspired by the movie Tron: Legacy by Disney displayed in Figure 1 below. Lume utilizes the dark and moody electronic sounds of the Tron universe as inspiration to create a unique ambiance for the player as they visit the different sections of the KOG empire. Visually, the user is presented with an initially dark and bland world that lights up in vibrant neon oranges, blues, and greens as the player harnesses more energy and interacts with different areas of a level. The building architecture mimics the rigid futuristic building style used in Tron but adds a unique organic texture style to the sides of buildings that the player interacts with.

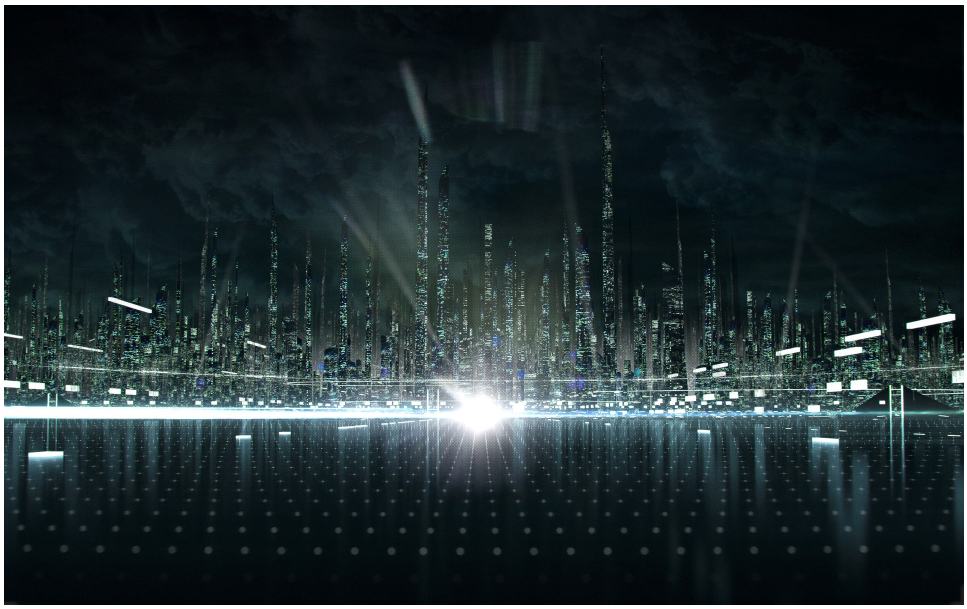

Figure 1: Tron: Legacy

Throughout the development of Lume, the concentration was focused on adding new innovations to the 3D platformer genre. Lume's game-play was inspired by several other 3D platformers including: Assassin's Creed by Ubisoft, Super Mario 64 by Nintendo, and Mirror's Edge by Electronic Arts. Traversing through the world by rooftop was a mechanic used in Mirror's Edge displayed in 2 below and Assassin's Creed displayed in Figure 3. The inspiration behind completing various objectives in the different KOG districts stems from the star collection mechanic of Super Mario 64 displayed in Figure 4. The key difference between these games and Lume is the innovative way in which the player traverses the world. As developers our intent was to create an intuitive path through the levels, however, with the ability to build on nearly every building in the KOG world the player is free to create their own path through a level. Minecraft was the main inspiration behind allowing the player to modify the world by building blocks anywhere in the world. Allowing the player the change the world freely grants them the ability to form their own path and visually alter a level differently on each play through.

Figure 2: Mirror's Edge

Figure 3: Assassin's Creed

Figure 4: Super Mario 64

Technologies

Lume was implemented with a variety of graphics technologies that allowed the game to have an aesthetically pleasing look, while running in an efficient manner.

Below is a list of the various technologies that were implemented and team member(s) that worked on them:

- 3D Interactive Environment (All)
- Collision Detection (Mike Buerli, Ryan Schroeder)
- Spacial Data Structure (Mike Buerli)
- Frame Buffer Objects (Mike Buerli)
- 3D Modeling and Animation (Teal Owyang, Brent Dimapilis)
- Model importer (Teal Owyang)
- “Spring-Loaded” Camera (Jeffrey Good)
- Freeform Camera (Ryan Schroeder)
- Camera Pathing (Ryan Schroeder)
- Robot AI/Pathing (Jeffrey Good)
- Bloom Shader (Jeffrey Good)
- Mapper/Importer (Jeffrey Good, Jonathan Rawson)
- Level Design (Jonathan Rawson, Jeffrey Good, Trent Ellingsen, Ryan Schroeder)
- Objectives (Ryan Schroeder)
- Growing Objects (Mike Buerli)
- Moving Objects (Jonathan Rawson)
- Heads Up Display/Main Menu (Jonathan Rawson)

- Picking (Mike Buerli)
- Player Logic (Ryan Schroeder)
- View Frustum Culling (Ryan Schroeder)
- Particle Explosions (Trent Ellingsen)
- Dynamic Abstract Lighting (Mike Buerli)
- Animated Textures (Trent Ellingsen)
- Textures & Logo design (Trent Ellingsen)
- Simple Level of Detail (Trent Ellingsen)
- Orbs Particle System (Brent Dimapilis)
- 2D Billboard Texturing (Brent Dimapilis)
- Robot/Plant Texturing (Brent Dimapilis)

Look & Feel

Within the game environment there are three 3D models that represent characters. These models were created using Blender and include the main character ‘Lume’ as well as two of the enemy robots controlled by the KOG corporation. The levels in which the game’s world resides were created through a stand alone map creating program. The mapper exports a custom file type, developed by the team, that the game is able to interpret and construct each of the worlds with. One of the technologies implemented in Lume was billboarding to simulate a 3D look using 2D objects. To create the look and feel of futuristic posters and logos, 2D textures were placed in 3D space and were alpha blended create the effect of glowing advertisements. There are two particle systems in place, the first occurs when the character gathers of energy and there are orbs that follow the character in a flocking simulation, the second occurs when robots are sprinted through and killed. Using both bloom and blur effects, the outer glow of the building was manufactured. Animated textures were used to create the title, intro, and loading screens.

Optimization

Lume implements several technologies that allow for optimized gameplay. To help the bottleneck of the graphics pipeline, Lume does not send all the geometry down the pipeline every frame. Using view frustum culling, objects that are not currently being viewed by the camera are culled out and not rasterized by the GPU. Skyline, a level in Lume, has a group of 70 robots. In order to continue ensuring smooth gameplay during Skyline, level of detail was implemented to draw objects with less geometry when the player is further away and objects with full geometry when closer. Each object drawn to the screen has an update function that is called during every frame. The objects are rendered this way in order to alter color or move blocks. The object update function is turned off when objects are too far away. To further improve performance a uniform

spacial data structure was implemented that breaks up the world into a 3D grid. The 3D grid allows for testing collisions based upon the character's position. The hit test function is only used to check the collisions of objects occupying the surrounding bucket spaces. To optimize the collision detection, Lume uses axis aligned bounding boxes to test the potential hits.

"Spring-Loaded" Camera

When playing a third person game of any flavor, the camera must behave as a player thinks it should. This means that it should not go into the walls or stop moving because it is being blocked. The solution that is utilized in Lume is a camera that zooms in if it detects collision with an object on the x-z axis of the y plane equal to the camera's y position(see figures A and B). After the first initial collision, the algorithm will then zoom the camera in repeatedly until it is no longer hitting an object. There is a counter that is kept in accordance to the amount of zooms done. It is maxed out at 100 zooms and if it is greater than 0 than the game will attempt to zoom out one level per a frame if it is possible, which is where the "spring-loaded" prefix comes from.

Figure A
Normal Camera

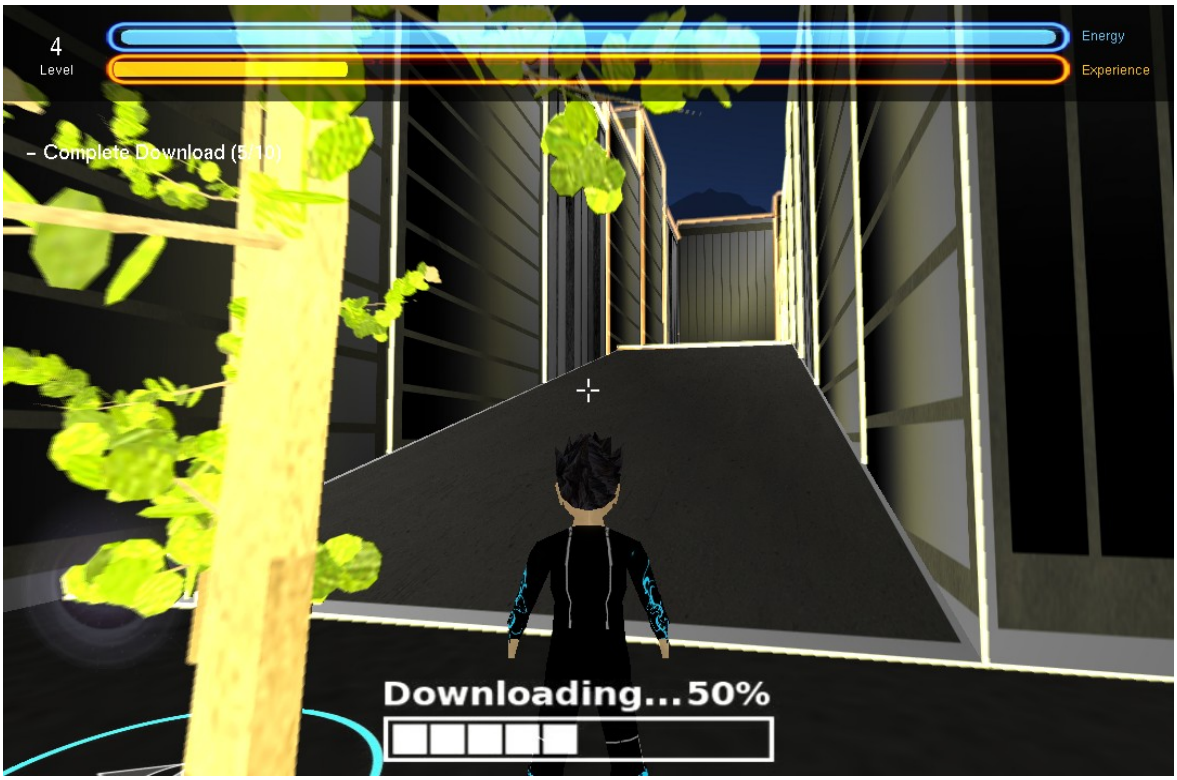

Figure B
Zoomed Camera

Robot AI/Pathing

In the first few builds of the game, the world seemed empty because it was just project Lume and the buildings. So in order to make the world feel a little less empty, Lume contains robots that patrol around their own designated area. A robot's path consists of a list of possible destinations which are read in from a text file. The robots are added one at a time based on the id given in the path file. If there is a duplicate id, then a path node with those coordinates is added to the existing robots set of destinations. As the robot reaches a destination it will go to the next destination in the list, and will iterate repeatedly through the list. To add a little bit more life to the robot, a sin wave was added on to the y axis. In certain levels, the robots will begin to chase Lume when he is within a certain radius of them. After leaving the radius the robot will resume its path by going straight to its previous destination node. The chase part of this algorithm is done by updating their velocity every second according to the difference between their position and Lume's position.

Mapper/Importer

Hardcoding levels into a game is not a good decision, because of the time and abstractness of humans using words to describe a 3D world. In order to more easily create good looking and functional levels, Lume has a dedicated 3D mapping program packaged with it. This solution allows for a layer of visible interactivity between the developer and the actual level architecture. The mapper runs as a standalone program and exports text files that are read in by the main game. The main unit used in the mapper is a brush, which is essentially a 3D cube. Each brush has an associated texture that it is drawn with, and a type that is used to determine how it will appear one loaded in the game. The general structure of the program is modeled after radiant, which is a 3D mapping program used for the Quake, Doom, and Call of Duty series of games(see figure C). It has three subwindows each representing information on the map in a unique way. The “grid” window is a 2D view of the map where brushes are created and modified by dragging on the grid. The grid has three views, x-y, y-z, and x-z, which allow for multiple views of the data that might not be available on one of the other views. The “preview” window gives the map creator a general sense of what will be seen in game by giving them a explorable 3D view of what they are creating. The “texture/type” window shows a preview of the currently selected texture, it also has a right click menu attached to it that allows the user to import new textures or select a type for a currently selected brush. As is necessary to making modifications to maps, there are import and export functions that save and load all the information necessary to continue mapping whenever needed.

In order to load whatever was exported from the mapper into the game, a map loading function was written and modified to suit the needs of the game. In its original incarnation the importer only loaded up the map file without any types and relied on the texture name to figure out the type. An old tutorial map was created as a test for the importer and was eventually phased out of the game completely as it outgrew original expectations(see figures D and E for old tutorial in mapper and in game). As the game evolved so did the importer, types were added and new files were added such as the robot paths, the 2D details on the wall, floating advertisements, and objectives. This versatility made it really easy to add new things to the world as they were being developed, and is still easily expandable for any new ideas Lume may incorporate in the future.

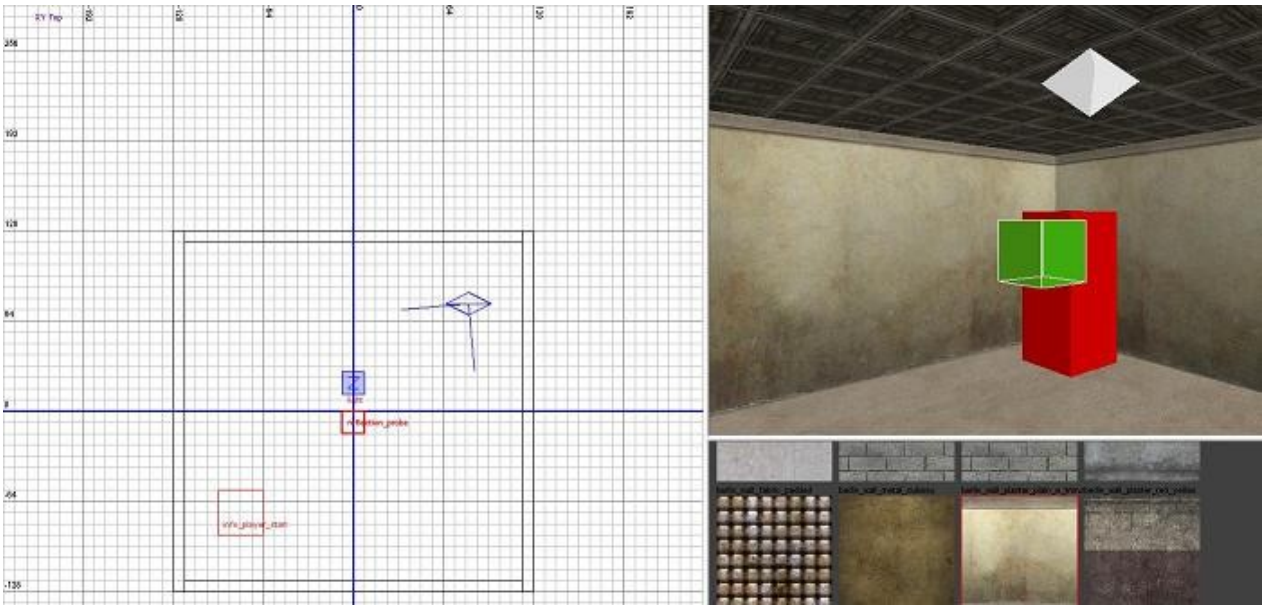

Figure C
Call of Duty Radiant

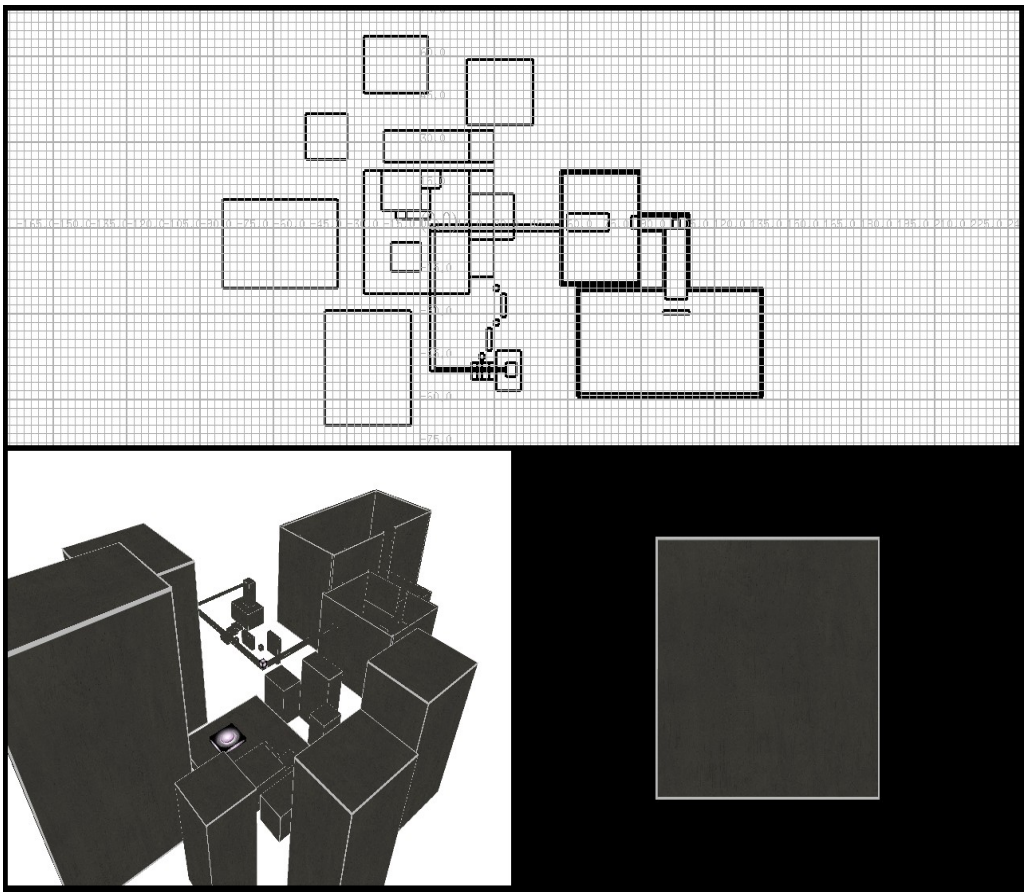

Figure D
Mapper w/ Old Tutorial

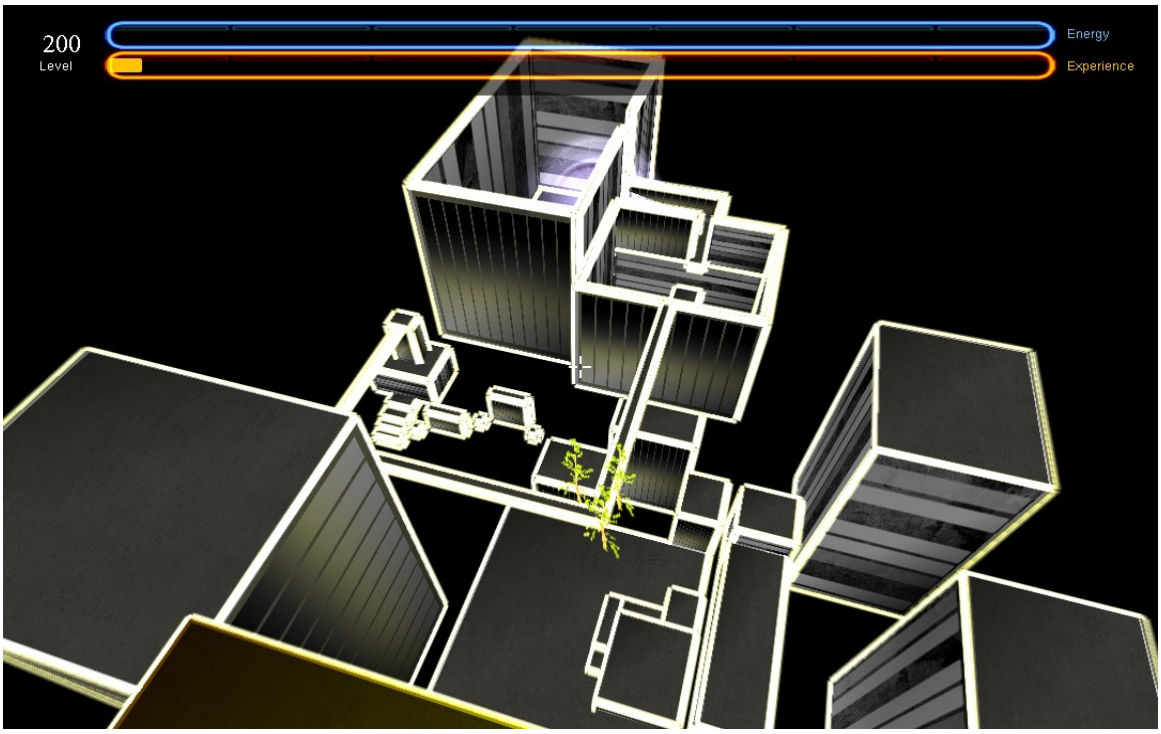

Figure E
Game w/ Old Tutorial

Save/Load

The game has roughly a three hour playthrough time for first-time players. Because of the length of the game, a save-load system was made to keep players from having to start over each time they play the game. It functions close to the same way the map exporter/importer does, except it saves key information on the current state of the game, rather than the map itself. Upon saving, in terms of the character, their experience, energy, level, position are all saved. In terms of the level, the current level and checkpoint are saved, all the growing objects and their 2D decals on the wall are saved, and drained status of all the checkpoints are also saved. In terms of more general states, the games objective statuses and trigger information are all saved. Upon loading, all this information is parsed and the game state itself is matched to what was originally saved out, allowing the player to play seamlessly through the game without having to keep it running when they stop playing.

Bloom Shader

Since light plays a key element in Lume, the effects of light on the world needed to look realistic, in order to do this a Gaussian blur effect was necessary. This effectively is the bloom shader, which takes a pixel and brightens it according to its surrounding pixels. Using the frame buffer object as a texture, the shader goes through it, pixel by pixel,

and tweaks the color to give the blurry effect of emanating light. The algorithm grabs the 4x3 box of pixels around the currently selected pixel and grabs an approximation of the average color of them in a 4D vector. After adding them all up, the red value of the current pixel is checked for its intensity. The more intense the red the less bloom will effect it and vice versa(the before and after results can be seen in figures F and G, respectively).

Figure F
No Bloom

Figure G
Bloom

Ghetto Level Design

A city is made of many districts, and this is no different in the city of KOG. The ghetto is the first of the four disticts that project Lume explores in the game. It was the first level designed for use in the game(the original tutorial level was more of a demo/test for the mapper and importer). Because sprint was not included in the game at this point, it was designed with only being able to use the growing objects as a means to traverse big gaps. The level was first drawn up on paper, then created in game using the mapper(see figures H and I). In designing the level traversal itself, a tightly wound level was designed, in which the player would cross over places they had already been, and by the end of the level the player would be able to see where they started. In regards to the placement of businesses and advertisement, fast food and soda were aimed for because it would be what one would see in the ghetto of a modern city.

Figure H
Ghetto on Paper

Figure I
Ghetto in Game

Suburbs Level Design

The suburbs is the third district visited in the game. It, in contrast with the ghetto, was the second to last level to be created, and thus was built with all finalized abilities in mind. This level, like the ghetto, was prototyped on a piece of paper, although this time around it was a very general design of the character's path(see figures J and K). As suggested by the name of this district, the suburbs is themed to be a residential area with houses and apartments. There are 4 towers in the level that each house an objective at the top, and each have huge billboards on the side. After getting the first objective the district lights up and all the moving platforms in the area are activated allowing access to the other 3 objectives in the map. The moving platforms require good timing on the players part to traverse and are probably the most challenging part of the game. Due to this being a residential district, there are no advertisements, but rather the player is treated to KOG propaganda on the towers around the district that remind residents that the ever watchful eye of KOG is present. The three apartment complex are each a letter KOG, which is more of an easter egg if the player notices, but does emphasize that KOG is everywhere, even where you sleep.

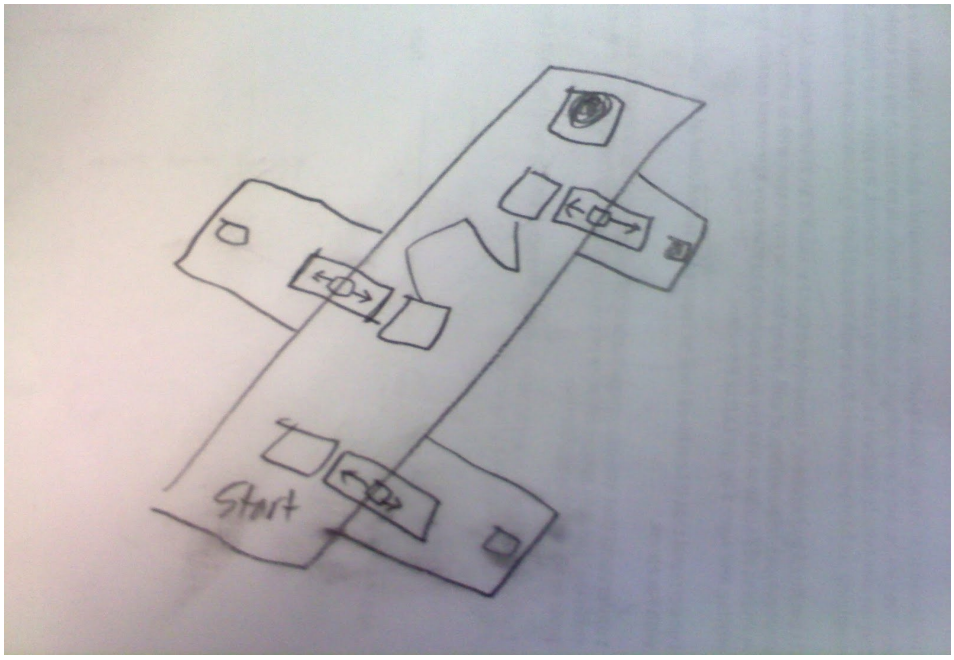

Figure J
Suburbs on Paper

Figure K
Suburbs in Game

Results

At the beginning of this project, all we had were some Youtube videos to give us some inspiration for the look and feel for the game, and some tools developed during the previous quarter. After two quarters, we were able to turn Lume into a playable game with a complete story and a unique look and feel.

Our team was particularly proud of how we were able to take our limited graphics knowledge, and creatively use technology to create an appealing look. After completing a level, the world lights up (Figure 5). Robot enemies are destroyed when sprinting through them, creating a particle effect that surrounds the robot. (Figure 6) 3D models of trees are animated to be brought back to life (Figure 7). A bloom shader is used on top of the trees to give them a pretty glow (Figure 7). An excessive amount of blocks built on a single wall shows the creation and life that is brought to the game (Figure 8). The look of the blocks on a wall is very appealing and demonstrates the freedom the player has to change the world he or she is in.

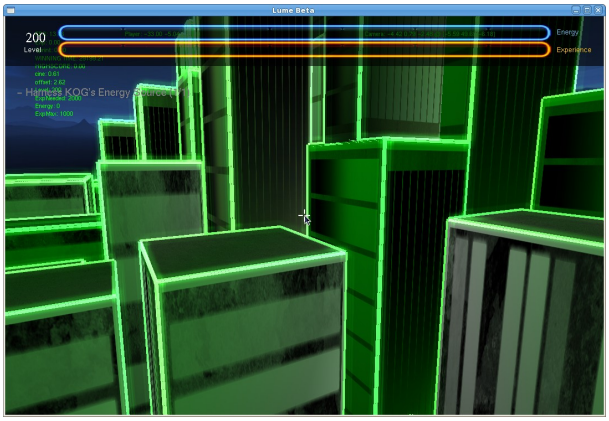

Figure 5

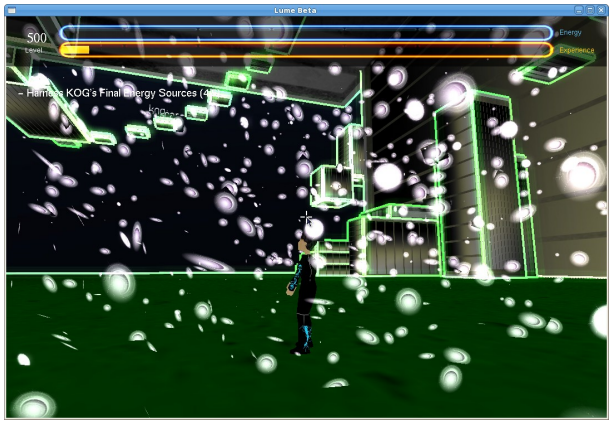

Figure 6

Figure 7

Figure 8

We were also happy that we were able to create a game with a simple game mechanic that gives the player the freedom to explore the world while still having set objectives that keeps the player interested. Below is a picture of the block that a player has built (Figure 9). The player can jump on the block to maneuver around the world. In the top left of the figure is the current objective the player must complete, with a progress bar towards completing the objective below.

Figure 9

Play Tester	Comments
Joanne Mark (Week 15/20)	**Fun Level (1-10)** **1- Not fun / 10 - Totally awesome** 8 **Game Crash?** No **Laggy?** No **Bugs seen?** Yes : Can see through roof w/ camera Camera goes into the wall Ground causing death should be more clear **Likable features**

	● Finding / Gathering insight ● The trees ● The energy trails ● The evolution from the gobj ● Outline glow of the buildings ● The music **Dislikes** ● Without building outlines it is hard to tell distance ● Didn't like the ramp to ramp jumps ● Don't like the last part of suburbs with up & over needed to go up the levels, better if it was just going up ● Need more check points **Story Interpretation** Save your city from evil KOG by getting all those energy light thingies **Suggestions** ● Have a girl character too! ● Have high score list (maybe have something similar to Zelda load screen where it shows stats) ● Multiple saves for different players ● Freebies - find hidden star to get extra energy / cannon shot / turn on moving walkways **Miscellaneous** Stopped at the back of the energy source building in the suburbs because of frustration
Brian Sukkar (Week 17/20)	**While playing Game Suggestions** ● Make objective more clear - Download insight isn't self explanatory ● Make controls more clear (maybe cut scene?) ● When first enemy appears tell about sprint to kill them ● Change level 1 block that is floating ● Full animation ● Suburb if go wrong way on first part then hard to go back. **Fun Level (1-10)** **1- Not fun / 10 - Totally awesome**

	6/7 **Game Crash?** No **Laggy?** at one point -> see bug section below (otherwise no lag) **Bugs seen?** ● Died for no reason (probably lag + enemy shove) ● Camera went all wack and couldn't see ● Feet off edge but still on block ● De synced moving platforms **Likable features** ● Pretty ● Challenge is fun ● Different routes are good ● Difficult to control direction at first but felt good after a while ● Everything moving **Dislikes** ● Jumping through blocks - it doesn't seem different enough once that feature is introduced **Story Interpretation** KOG took world - you kind of bring life back **Additional Suggestions** ● Fit trees more (but cool as is as well) ● Like the give life but maybe make purple? like Avatar - the movie **Miscellaneous** Maybe black hole or something to bring to beginning / teleport at top of level
Ken Li Week (17/20)	**While playing Game Suggestions** ● Ability to go anywhere makes me feel unsure if I'm going the right way ● Thought skybox was a wall and jumped to death

	● Asked what the 2/10 was (didn't recognize that it was an objective) ● Camera zoomed in when walking along wall ● Want animation for creating the blocks (referenced the portal gun) ● “Defeat KOG” in overworld was unclear and thought I didn't complete 1st world (maybe it could come up after other worlds) ● Can fall in a place w/o energy and get stuck (maybe use 'r') ● Can't feel effects of the +1 to abilities ● NEED MOMENTUM TO CARRY AFTER SPRINT RUNS OUT ● Frustrated on suburbs because of the lack of energy ● Walk animation odd ● In suburbs, explain moving platforms will be turned on once you complete the first objective **Fun Level (1-10)** **1- not fun / 10 - totally awesome** 5 **Game Crash?** No **Laggy?** No **Bugs seen?** ● Leveled up then able to take all blocks back ● Bad picking sometimes (character against wall trying to remove block placed to the side) **Likable features** ● Graphics ● Colors of making blocks ● Likes building lighting up when close ● First level well made **Dislikes** ● Frustrated ● Check point so far back

	• Save feature needs explanation **Story Interpretation** Parody version of Tron **Additional Suggestions** need 'e' explained suburbs not well built likes if there is narration Would like some sort of gun that shoots the blocks or some indication that he has the power **Miscellaneous** Stopped out of frustration on bottom of suburbs tower. Took portal into first, then left to unlock the other worlds.

Based upon earlier feedback from classmates and demonstration viewers, certain aspects of Lume were changed or expanded upon. One of the design aspects was to create a better way to show that the setting is a city. To accomplish this, billboards and logos were designed and integrated into the futuristic city (Figure 10).

Figure 10: Logos in City

Another aspect that was commented on was the lack of interaction with enemies or other characters. Users wanted another way to act against the robots of KOG. In

addition to the “sprint through to destroy” interaction, the ability to “freeze” large robots was added, to give the character the ability to feel more powerful. This freezing ability is shown in Figure 11.

Figure 11: Freezing Ability

This experience gave the development team invaluable insight into what it is like to make a game on a team. Each technology that was developed had a unique way to make the game better, and we used each team member’s strengths to make a positive contribution to the game. The game would not be the same without the effort of each team member. The tasks that each person completed were motivated by decisions made about game mechanics, aesthetic appeal, story, technology, and play tester opinions. Our game was played by students from Cal Poly, including computer science masters students concentrating in computer graphics. Having fresh eyes playing our game was a great help. The play testers were able to convey what didn’t feel right in the game, explain what they liked about the game, and describe what was not clear in our game. Fixing these problems or expanding upon things that people enjoyed allowed for the team to make a game focused on the player’s desires.

The game testing started in the later stages of our game development, so the game mechanics and most of the look feel had been determined by this time. The major contribution that game testers gave were comments that clarified the story. Some players were confused at what the story was, and we later created cut scenes at the beginning and end of the game to clear up the story line. Game testers also said they

felt lost at what to do in the game because it of their ability to roam freely. To address this, we created clear objectives in the game that gave the player an idea of what they are supposed to do in each level. We were also constantly modifying individual levels in the game to have a linear amount of difficulty as a player progresses through the game.

Dividing the work was done based on each member's strengths or what they were particularly interested in. Allowing everyone to choose what they were interested in resulted in a more rapid development because each member was eager to learn about the technology involved. Small teams were assigned to different tasks in order to prevent anyone from working alone. These teams made up of two or more members allowed for more monitoring and motivation for each member.

Working alone as oppose to working in small teams was shown to be inefficient during the two quarter process. Deciding on design decisions without the input of at least one other member was likely to result in individual error. Furthermore, code was harder to interpret and errors were harder to debug when other members looking at it were not directly involved. Overall, our development process can be described as allowing small teams to rapidly develop technologies that they were most interested in, while in a manner that utilized each team member's strengths. This was proven to be successful as is indicated by the progress of game over these last two quarters.

Conclusion

Making Lume has been an amazing experience, from the team I worked with to the final product, it all came together better than I could have imagined. From a software engineering standpoint, it was a great experience to work with a big and diverse team. We made decisions as a whole on the vision of where Lume was going, which led to minimal conflict between the interest of team members. From a graphics standpoint, it was amazing to see the differences that new effects, textures, and cinematics had as they were added into the game. It was also really helpful to have the experience of implementing a few different 3D environment/graphics optimization algorithms, which were all needed when frame rates dropped as the game got more complex.

From a general programming standpoint, this was the biggest project I have been a part of, so managing the code and making sure all the components worked together was key. Creating a file format for shared use between the mapper and importer is a good example of this, because that file is the only way that the game and the tool can communicate between each other. This was also the first project in which I have really done a lot of C++ in, so I now consider myself an advanced C++ user upon its completion.

Future Work

After two quarters of work, Lume has become a fully fleshed out interactive 3D game. The next stage, should the team choose to pursue it, would be preparing Lume for release on Steam. The following aspects of Lume would need to be completed in order to confidently release Lume on Steam for users around the world to play.

More In-Depth Storyline
Currently, Lume has a complete storyline which is not conveyed clearly enough through playing the game. To improve upon this, the team would need to
further develop the storyline by including more artistic cut scenes and more objectives that add purpose to the player's choices. Interactive communication with Insight, so that the player can ask questions, would also expand upon game mechanics and story elements.

Memory Management
None of the classes in Lume have taken full advantage of destructors in C++. In addition to running a profiler to determine which methods can be optimized for better performance, the implementation of destructors for all of our classes would greatly improve the performance of Lume on older machines.

Portability
The current system specifications for Lume are 32-bit Linux operating systems. In order for our game to have the impact we wish for it to have, it must be modified to be playable on Windows and Mac systems. This requires some redesign of the code, most notably in the included libraries in each header file and the Makefile. Some function calls are specific to the operating system (namely Windows) and would need to be changed. This would allow us to distribute our game on Steam for any operating system.

References

Subwindow tutorial used for mapper.
http://www.codeproject.com/KB/openGL/glut_subwindow.aspx
Call of Duty UO Gradient inspiration for mapper
Bloom Shader tutorial
http://kalogirou.net/2006/05/20/how-to-do-good-bloom-for-hdr-rendering/