CORE 13

Jereis Zaatri
CSC 491
Project Advisor: Zoë Wood
Spring Quarter 2011

Table of Contents

Introduction	003
Project Overview	004
Related Works	005
Algorithm Overview	007
Algorithm Details / Design	008
Results & Future Plans	011
Screen Shots	012

Introduction:

In the past, starting any company was a costly ordeal as all aspects of business involved a great deal of cost from production to sale. For instance an author would need the equipment to write the book, an editor to proof read the book and a publisher to distribute the book. With the advent of the digital age, getting content out can all be done behind a computer with little to no cost.

Among the many forms of entertainment, the gaming industry is becoming a prominent industry rivaling that of the movie industry. Today, prominent publishers like Electronic Arts (EA) are making headlines with game titles such as Dragon Age and the Battle Field series. Working with such prominent firms has its conveniences, but as an independent developer, there is a lot more control over the production and ownership of the intellectual property. Plus, as an independent developer, name recognition isn't a necessity like other studios that need financing from such publishers. As leader of the Core 13 project, pursuing this entrepreneurial spirit is at the root of this project. During the production of Core 13, every element of game design, from editor to game physics was explored, leaving no element outsourced to an external library or application.

The Core 13 project is composed of a 3 member, Seth Black, Josh Maass and Jereis Zaatri. The project is led by Jereis Zaatri and was produced in the span of two quarters. The objective of the project is to produce a game from scratch with the possibility of marketing the game through on-line retailers. Progress of the project, oversight and approval of the project was done by the magnificent and all powerful, professor Zoë Wood.

Project Overview:

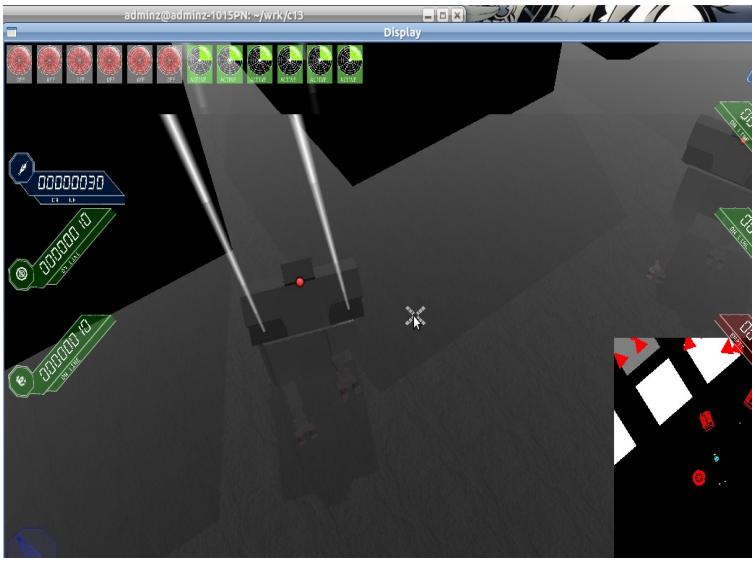
Core 13 is an arcade styled mech battle game. Unlike other mech games, rather than emphasize realism with a slow and clunky metallic giants, the player is provided a fast paced battle that leaves no time to idle around. As fun as it sounds to pilot a giant robot tearing down its surroundings, such games fall short when the players patience is tested. At heart, such games are just first person shooters which requires a fast paced game play. On a more technical side, the Core 13 project is as much a game as it is a technique for making a game. The project includes a level editor and model builder built from scratch. Everything down to the texture was built from scratch with audio being the only exception. This project was done with cross platformed libraries (GLUT, freeGlut, SDL, etc), allowing the game to be built for mac, Windows and Linux.

As for the people behind the Core 13 project, a team composed of two computer engineers, Jereis Zaatri and Seth Black and a computer science major, Josh Maass started the project. The origin of the game title comes from the fact that the player needs to manage energy cores as means for the player to choose between being powerful yet vulnerable by enabling all core or slow and safe, by enabling a few cores. As for the number 13, the F-keys were going to serve as a means to toggle the cores and at the time of the title creation, the team leader Jereis Zaatri thought there was an F13 key.

The idea for the game stems from a child hood interest in huge battling robots and seeing large explosions. Unfortunately the ideal mech battle hasn't been made yet. Most end up being too slow or too short of a game. As for the story line behind the game; you, an average guy from the future, a future where the old cities are no long inhabitable, wars are thing of the past, the entertainment industry is booming and a new form of

entertainment has emerged. In the future, people entertain themselves by battling it out with remote controlled decommissioned military hardware like drones and mechs. Rather than play in some virtual world with virtual characters, players duke it out in these uninhabitable cities that need to be leveled for new development and what better way to tear down a decrepit city than with giant military hardware. For now the game will focus on the game play, but as this project is furthered outside of the academic realm, the story line will be emphasized.

As for the gameplay, power management is the driving theme of the game play. Players can choose to to go all out and enable all cores, which leaves the player more susceptible to damage or few to be slow and protected. In addition, the cores can be turned into weapons by launching them at you enemies for massive damage.



Related Works:

Mech games aren't new, but they haven't reached a mainstream status as other genre of games like role playing games like Final Fantasy from Square Enix or first person shooters like the Halo sereis from Microsoft. As mentioned in the overview, mech games often fall short for one reason or another. For instance Steel Battalion, published by Capcom is quite the eye catcher with is HUGE controller (figure 1), but with it's $200 price tag and simulation like controls, it wasn't a game for the mainstream audience. If there's anything that'll scare off a mainstream audience, it's a huge price tag

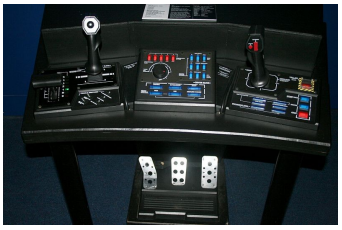


Figure 1: Steel Battalion controller

and a complex controller.

Phantom Crash (Genki) | Source: ign.com

On the flip side, there's games like Phantom Crash by Genki. Such games are a blast to play. They're fun, a quick game to pick up and fast paced. Unfortunately Phantom Crash suffers from one key short fall, it's too short. With only three level and so many ways to upgrade your mech, eventually the player exhausts every option in the game.

As for creating the game engine, besides being a course requirement there are benefits over using an existing engines. In this case, the Torque Game Engine will be used as a comparison as it is a familiar game engine to the group leader, Jereis Zaatri. Torque uses a scripting language called Torque Script which is an object oriented language, similar to Java. Unfortunately there are a few short falls to using a scripting based engine. Overhead is often a short fall of such engines. With so much going on in between the native machine code and the abstracted scripting language, there is a vast window of error that can take place. Debugging often becomes a hassle under such platforms as a large gray area exists between the script language and the machine code where a bug can exist. Particularly in the case of Torque, the developer is provided both the script level and source code of the engine. Development on such a platform becomes a juggling act as both sides of the spectrum are managed by the developer. In the end, using such platforms becomes a trade off between platform independence and control.

In contrast to Torque Script, developing a game engine entirely in native machine code eliminates the “smoke and mirrors” of a higher level abstracted language. In addition, no overhead needs addressing to add functionality from the native level as the entire engine is developed at that level. With the absence of such abstractions, the developer has a clearer view of all the components of the game.

Algorithm Overview:

Below is a list of algorithms and relate work done each group member.

- Game Engine Cycle: Jereis Zaatri
- Physics & Spacial Ordering: Seth Black
- Game Structures: Jereis Zaatri
- Fog Design: Josh Maass
- Animation Specifications: Jereis Zaatri
- Mech Leg Animation: Josh Maass
- 3D audio: Jereis Zaatri
- Shaders & Shadows: Seth Black
- View Frustum Culling: Jereis Zaatri & revised by Seth Black
- Display List: Jereis Zaatri
- Alpha 32 bit BMP Image Reader: Jereis Zaatri (modified version 24 bit Bit Reader by Zoe Wood)
- Map Bounds Animation: Josh Maass
- Line of sight: Seth Black
- Map Editor: Jereis Zaatri
- Model Editor: Jereis Zaatri

Algorithm Details / Design:

- Game Engine Cycle

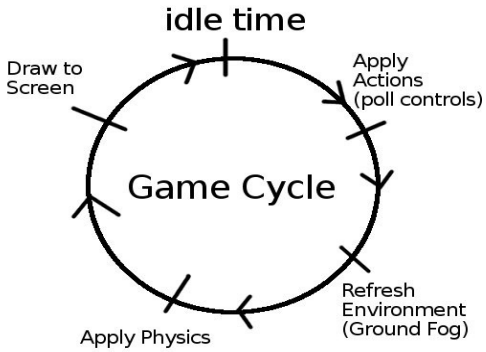


Figure 2: Game Cycle

Looking at figure 2, there's doesn't seem to be a lot going on and it's often the over looked due to it's simplistic nature, but there's important things implied by the design. Notice that the controls are polled, which significantly effects performance. Storing events and polling them later prevents the game cycle from being held up by a build up of redundant events in the event queue. In the early builds, this problem was observed and resolved.

- Game Structures

A central design was chosen over a modular design when developing the game engine. There is both a coding benefit and an organizational benefit to a central design. With only a single global structure to query information, far few headers have to be included and all game state information originates from one place.

- Animations

Like any other form of content, animations would have been very time consuming to produce. For this reason, animations were streamlined to an algorithm. In addition, since a mech is nothing more than a giant robot, a mechanical motion best suits the animation of a mech, which is what an algorithm is ideal for. The origin of this algorithm originates from looking at the motion of a cyclist. The feet of a cyclist move along a circular motion around a

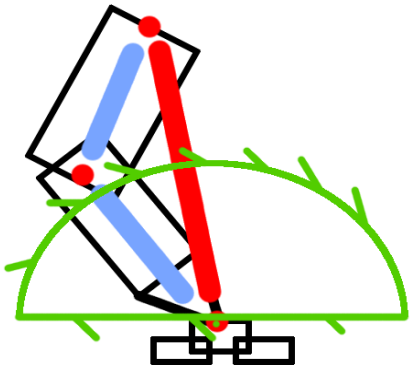


Figure 4: mech leg animation

gear with a diameter less than the distance between the hip and knee. By calculating the distance produced by the ankle and hip, and knowing the distance between ankle to knee and knee to hip, the angles for each segment of the mech leg can be determined. Such algorithms can also be applied to arms as well. In regards to using this algorithm for animation, since a sine and cosine wave can be used to produce the x and y position of the circular motion of the ankle; all that's left is to apply the algorithm to find orientation of each of the leg segments. Figure 3 and 4 is a visualization of the computation done to calculate the leg position. First, find the position of the ankle as shown by the green circle using sine and cosine. Next find the distance between the angle and hip as shown in red. To find the remaining two angles in green, determine the angle the red line produces with the vertical line and with the two know leg segment lengths (in blue) and a little trigonometry, all the necessary information is produced.

Figure 4: leg animation

- 3D Audio

3D audio is produced by separating stereo audio to left and right audio streams using an audio application. To find the appropriate ratio of left to right speaker volume, the horizontal angle between the source and player is calculated. From the angle, the ratio is determined. From this point the distance is determined to find out how much the audio is faded. Once these two calculations are done, the SDL mixer library is called with the needed audio streams and the desired volume levels.

- View Frustum Culling

One of the greatest bottle necks to drawing a virtual world is drawing geometry that doesn't show up on screen. By filtering out unnecessary geometry, the world can be rendered at faster rates. The view frustum is the volume that your eye sees. Figure 4 is an illustration of a dot inside (B) and outside (A) of the view Frustum. The process for producing the algorithm involves producing the plainer equation of each of the six sides of the frustum. Once obtained, points can be applied to the plains to check what side they're on. Since the normals and the plainer equations are closely related, figure 5 shows view frustum culling in the context of vectors.

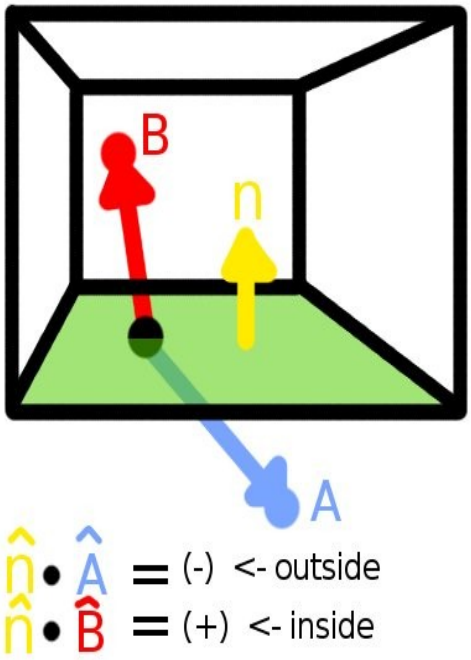


Figure 5: view frustum of bottom plain

- Map Editor

Shown in figure 6 is a map loaded into the map editor. Editing is done on a plainer based system. Rather than deal with the complexity of three axis of motion, the editor provides greater control with two axis of motion and three camera orientations to pan the camera with.

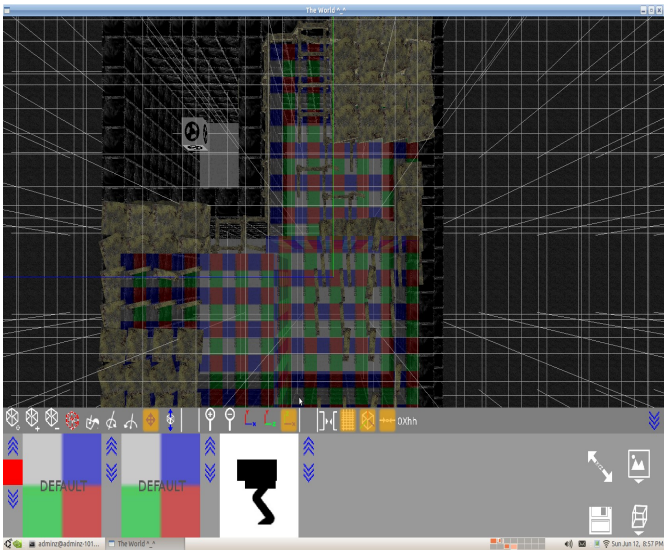


Figure 6: Map Editor

- Pseudo Models

With a limited amount of time, content had to be generated in a rapid process. The solution was to develop a “pseudo model,” a very light weight model that only stored texture points and static geometry. The idea behind using a pseudo model is that complex geometry can be reached by means of combining groups of simple geometry. In addition, the precision of the bounding box bordering increase as fewer gaps are take up by the bounding box. Figure 8 demonstrates this advantage, by showing a 3D V composed of one model and of two models.

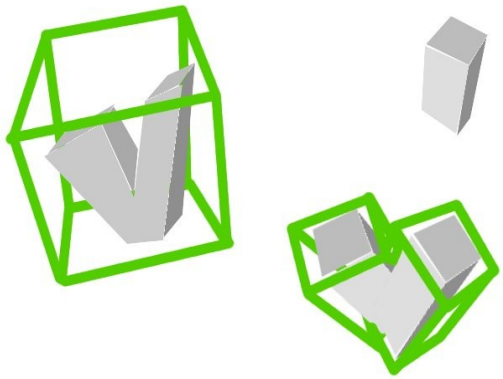


Figure 8: advantage of simple models

Results, Conclusions and Future Plans:

From an implementation stand point, the project is a success. Game content was able to be streamlined enough for two man team to continue the project. Most of the requested items made it into the game. Of the items remaining in the project as of the writing of this report, a story line hasn't been fully established and AIs still lacks a personality.

As for the outlook from a group cohesion standpoint, things haven't gone as planned. A few lessons learned from this project are not to assume your team mates will come into the project with the same level of enthusiasm for the project and that your team mates may not be the most social people on earth. For anyone planning to take the role of leader, be prepared to be a bit forceful as some members will likely not speak up when tasks are assigned. When the project was started, the project was left open for team members to provide their own input. Ultimately as team leader, gaging the social nature

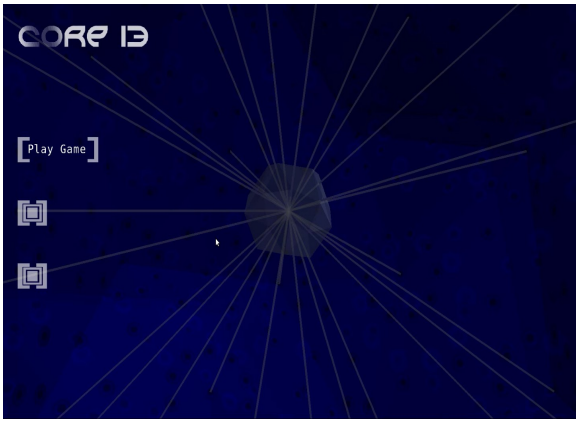
of the team and accommodating to it is key for success. If the team is too passive, a more aggressive and rigid approach must be taken, on the flip side, if everyone comes to the table with their own ideas, be prepared for some flexibility as disenfranchising team members is a formula for failure.

In regards to future plans, the game is going to be improved and the code will eventually be adapted to serve as a game engine for future games. Among the improvement, the AI will be upgraded to react to the current game state as well as interact with the player though preset dialogs. Graphically, more details will be worked into the mech and world design. An on-line component will be added and lastly a custom controller will be added to the game (see screen shots).

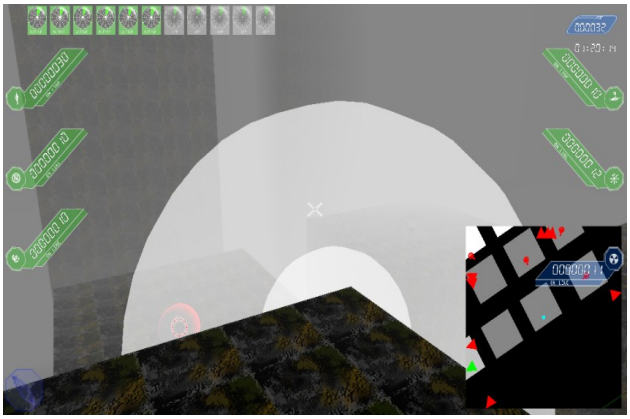
Screen Shots:

Enjoy the screen shot of the game.

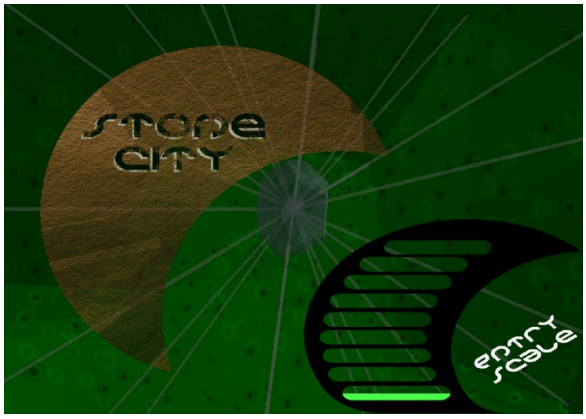
Logo Screen



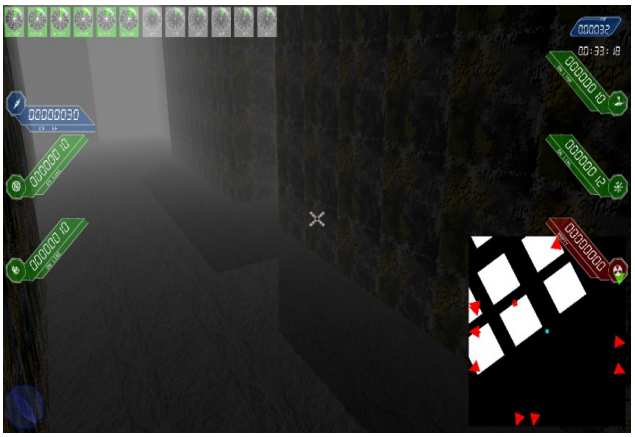
Main Menu



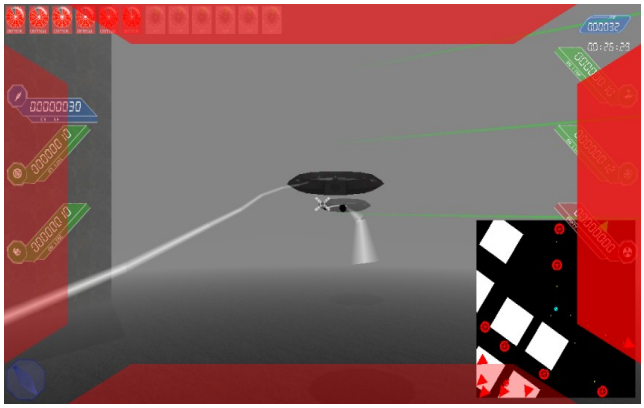
Exploding Core



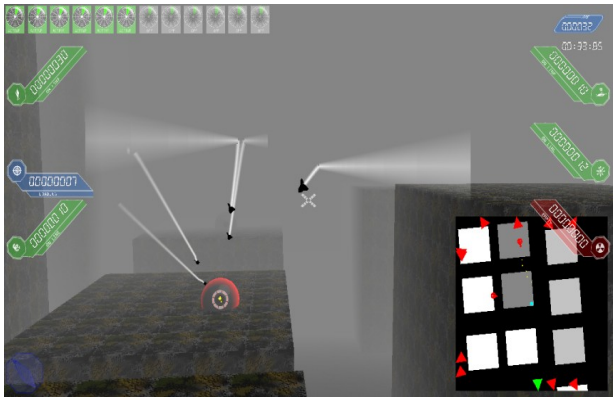
Level Select Menu



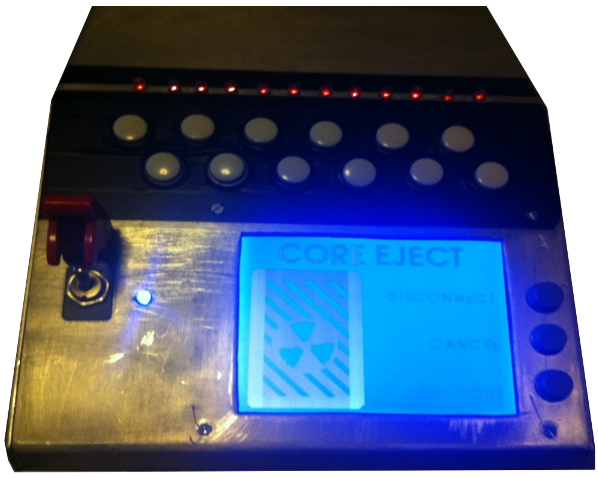
Enemy Mech



Enemy Hover Drone



Enemy Turret



Custom Game Controller