Project Number: RL1-P108

Hooping: a 3-D Space Racing Game

A Major Qualifying Project Report
submitted to the faculty of the
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Bachelor of Science

By

Sarah Gilkey
Marquis Pendleton
Dennis Valovcin
Tim Volpe

Date: March 17, 2009

Professor Robert W Lindeman, Major Advisor

Abstract

Hooping: a 3-D Space Racing Game

By

Sarah Gilkey, Marquis Pendleton, Dennis Valovcin, Tim Volpe

Hooping is a three-dimensional racing game that takes place in the vast expanse of outer space. For this project, our group designed, implemented, and tested a fully functional single player race game.

This report is a summary of the development of this game, from initial design through the implementation process. During design, we mapped out the high level design for the game, such as the control scheme for the game, game modes, and the ship selection method. We also discussed the specifics about the user-interface layout and different obstacles we wanted for the game. All of the design is discussed, and concept images are included.

During implementation, we learned a great amount about Terathon Software's C4 engine, causing us to rethink parts of the original design. With these modifications, we were able to fully implement a working game. We discuss the functional design of the game, as well as the implementation of all of the artistic assets in the game.

Acknowledgements

We would like to thank Robert Lindeman for advising this project. For his guidance and advice with project management and knowledge of C4, he was able to greatly aid our group. We could never been able to create this game without his direction and leadership.

Also, we would like to thank the team of Petrified, Chris Drouin, Skyler Clark, and Matt Fabian, for their constant advice and perspective while we were creating this game. Their insight and ideas were invaluable to the creation of out game.

Finally, we would like to thank the C4 engine forum community at http://www.terathon.com. Their advice and answers were indispensable to solving problems that arose during our project implementation.

Table of Contents

List of Figures

List of Tables

1. Introduction

This document describes the details of the project *Hooping*, as designed, implemented and tested in the academic year 2008-2009. The design is divided into three categories: game play overview, artistic design, and technical design. This document has been written as accurately as possible with the knowledge the team has at the time.

1.1. One Sentence Description

Hooping is a three-dimensional space racing game where players are challenged to maneuver customizable ships to fly through hoops along a given course while avoiding obstacles that can hinder their progress.

1.2. One Paragraph Description

Hooping is a racing game that takes place within the confines of outer space. Players are challenged to fly through hoops along a given course; however, there are obstacles between each hoop to test the player. *Hooping* offers players the ability to fly freely around tracks, the challenge of trying to complete a race on a given course as quickly as possible, and the challenge of completing a course before time expires. Players can customize an array of attributes of the ship to suit their tastes.

2. Gameplay Design

2.1. Game Modes

In *Hooping*, the player has the choice of selecting in which mode to race. There are three options: Practice, Time Trial, and Lap Count. The race mode is first selected from the main menu screen as shown in Figure 1.

Figure 1 - Main Menu Screen

2.1.1. Practice Mode

Practice mode allows the player to move through the course with no time limits or lap counts. This is primarily for game testing and player exploration. This gives the player an environment in which to get used to the controls as well as a way to explore the maps to become familiar with the courses.

2.1.2. Time Trial Mode

Time Trial mode gives the player a certain amount of time to travel between hoops. A time bonus is granted when the player passes through a hoop. If time expires before the player completes the course, the race ends.

2.1.3. Lap Count Mode

Lap count mode allows the player to set a certain number of laps to be completed for the race to end. The total time the player takes to complete these laps is shown to the player upon completion of the race.

2.1.4. Difficulty

Players can select Easy, Normal and Hard Difficulty maps when choosing the course in the game set up. These maps will vary in length, number of hoops, and variety of obstacles in each course. The more difficult maps will be longer with more obstacles to challenge the player, while shorter maps with simpler obstacles will be easier courses for the player to maneuver.

2.2. Ships

There are two ship bodies for the player to choose from during the ship selection process. The ships are designed as "stock" ships, which are able to have different parts attached to them. The player is given a number of credits to spend on four upgradable attributes: Speed, Acceleration, Handling, and Durability. Each change made in the attributes is shown visually by the associated picture on the right of the ship customization window (Figure 2). All of the ships have parts that attach in the same place on the body of the ship, regardless of which body the player chooses.

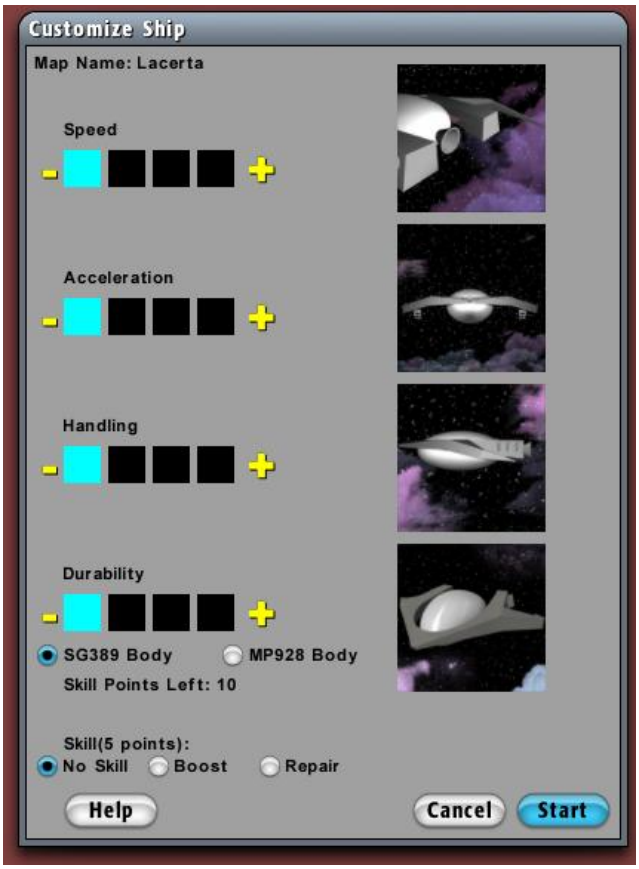

Figure 2 - Ship Customization

2.2.1. Ship Designs and Collision

There are two stock ships designed for players to use. These ships will be visually unique, but use the same base attributes. Each of these ships also require the different upgraded parts to attach in the same place in relation to the origin, in order for the parts to connect correctly to all of the stock ships. The concept design for ship bodies can be seen in Figure 3.

Figure 3 - Ship Concept

2.2.1.1. Damage and Death

During a race, if the player's ship collides with an obstacle while traveling with over 20% throttle, the ship takes damage, which is reflected in the ship's Durability Meter. If the ship is traveling with over 60% throttle, the ship's Durability is immediately brought to 0%.

Should a ship's Durability Meter reach zero, the ship is "destroyed". When this happens, the ship's current speed is set to zero, the throttle is set to 0%, an explosion sound effect is played, and the ship is repositioned to a safe location.

There is no functionality or performance penalty for taking damage; rather, the penalty comes in the loss of time as the ship comes to a standstill.

2.2.2. Ship Selection

During the ship selection process, the player is able to customize the ship to use during the race by adding or removing points for the given attributes (Figure 2).

2.2.2.1. Points System

The player is assigned a number of points to spend on customizing the ship. Each attribute of the ship has four levels of upgrades, each with a cost of one point. The ship starts with one point in each of the four attributes. As the player adjusts the attributes of the ship, a related picture of the affected part of the ship is updated to the right of the attribute.

2.2.2.1.1. Speed

The Speed attribute is a modifier applied to the ship's current speed. At level one, the ship travels 70% as fast as that of level two, and at level four travels at 150% of this speed.

The Speed attribute is shown visually through the upgrading of the engine, which is mounted on the back of the ship. The engine becomes larger and more powerful as the Speed attribute is upgraded as shown in Figure 4.

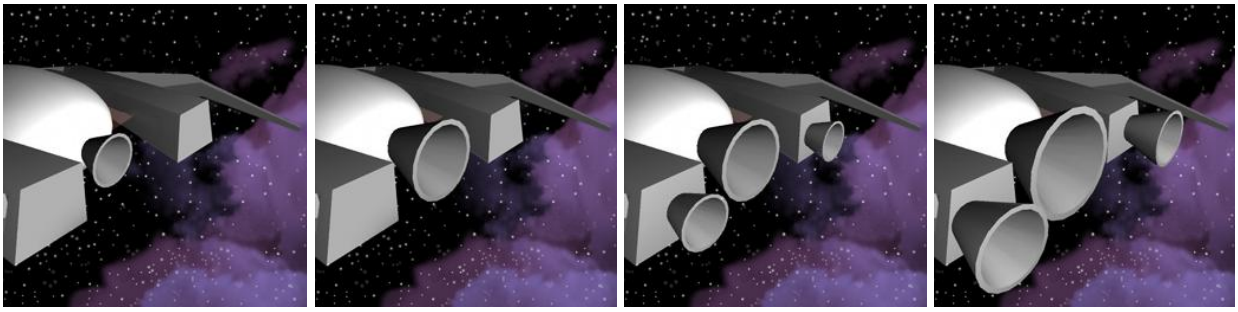

Figure 4 - Speed Upgrades

2.2.2.1.2. Acceleration

Acceleration is how long it takes for the ship to change its speed when its throttle is changed. Internally, this is calculated as the number of frames needed to increase the ship's speed by each throttle level. At level one, the ship takes 1000 frames to accelerate each 20% of throttle, while at level 4 it only takes 10 frames.

The Acceleration attribute is shown visually by the upgrading of the boosters mounted on the wings of the ship as seen in Figure 5. The size and number of the boosters increase as the Acceleration attribute is upgraded.

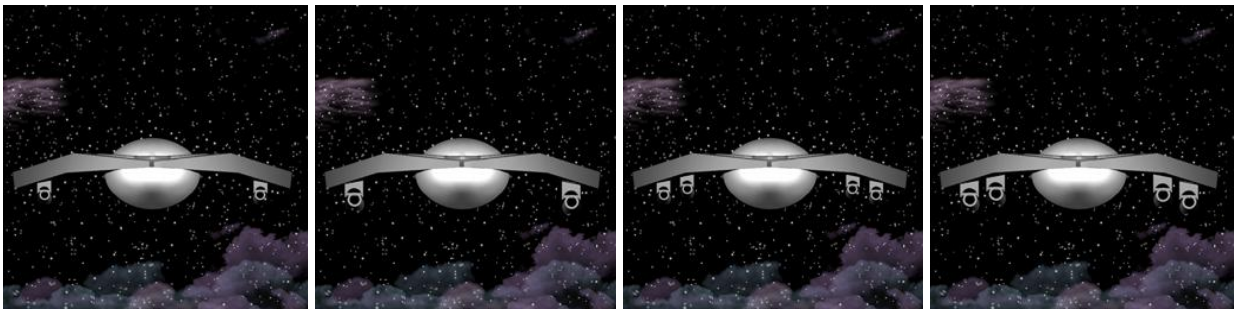

Figure 5 - Acceleration Upgrades

2.2.2.1.3. Handling

Handling is how fast the ship rotates when it turns. This is represented as a percentage of how much the ship turns per frame of animation. At level one, the ship turns about half as fast as that of level 3, and turns 130% as fast at level four.

The Handling attribute is shown visually by an upgrading of wings, which are mounted on the sides of the ship as shown in Figure 6. These wings become larger as the Handling attribute is upgraded.

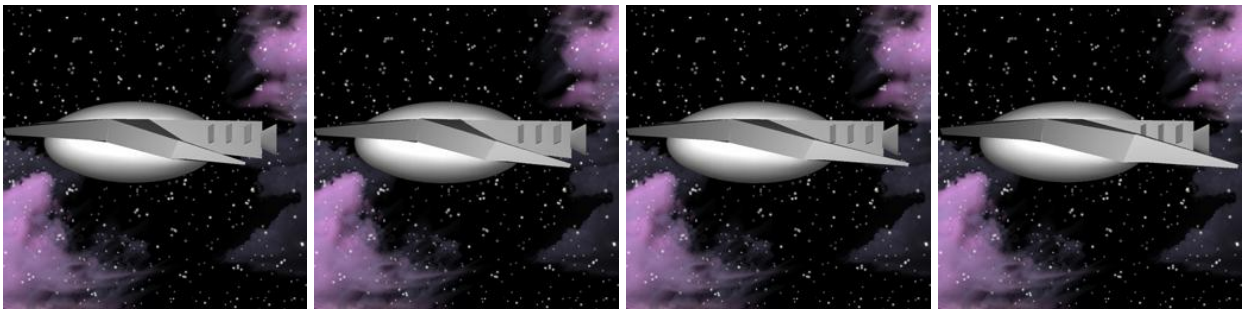

Figure 6 - Handling Upgrades

2.2.2.1.4. Durability

Durability indicates how much damage the player's ship can take before it is destroyed. The ship takes damage from colliding with the obstacles in the course. This is measured as a percentage of damage taken from collisions. A level one Durability takes full damage from a collision, depending on the obstacle, while a level four Durability only takes 70% of this damage.

The Durability attribute is shown visually by the upgrading of the ship body design. The ship body has more metal plating added when the Durability attribute is upgraded, making the ship look stronger and more reinforced an example for one of the ship bodies is shown in Figure 7.

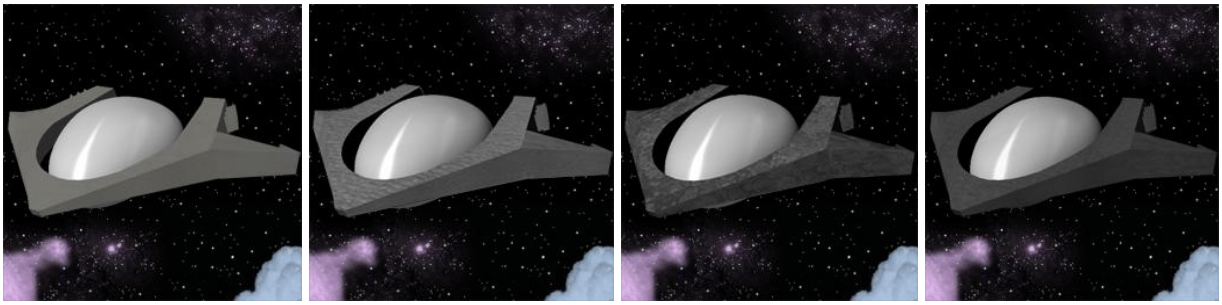

Figure 7 - Durability Upgrades

2.2.2.2. Purchasable Skills

In addition to upgrading attributes, the player has the option of purchasing additional equipment, referred to as Skills, for the ship. These include Boost, which propels the ship forward faster than its maximum speed, and Repair, which restores the ship's Durability Meter mid-race. These Skills have a significantly higher point cost than the attribute upgrades.

2.3. Controls

Input refers to the controls the player uses to interact and play with the game. An effort was made to design the controls to be as intuitive as possible for the player. Many games have common keys that are used and are familiar for the player and this was kept in mind for the control scheme of *Hooping*.

2.3.1. Keyboard controls

The ship's speed is controlled by a throttle-based system. The player presses the Accelerate (default W) or Decelerate (default S) buttons, which adjust the throttle in increments of 20%, in a range of 0% to 100%, where 100% is the ship's maximum speed, as determined by its Speed attribute. The time it takes the ship to accelerate or decelerate to these levels is determined by the ship's Acceleration attribute.

For example, a ship's maximum speed is 100 units/second. Pressing Accelerate moves the throttle to 20%, so the ship accelerates from 0 u/s to 20 u/s. The ship then continues forward at 20 u/s until the throttle is adjusted again or the ship hits an obstacle.

A Skill Use (default Space Bar) button activates the ship's Skill, if one has been selected during the ship customization process.

2.3.2. Mouse controls

The mouse is used as a standard pointer during game setup. This includes any menu screens before the game as well as the option menu during the game.

During a race, the mouse acts similar to a joystick, utilizing a rate-control system. Pushing the mouse left or right away from center yaws (rotates along the Y axis) the ship left or right until the mouse is re-centered. The speed that the ship rotates is a percentage of how far away from center the mouse is, based on a maximum determined by the ship's Handling attribute.

Pushing the mouse up or down away from center pitches (rotates along the X axis) the ship up or down until the mouse is re-centered. The speed that the ship rotates is a percentage of how far away from center the mouse is, based on a maximum determined by the ship's Handling attribute. In other words, a higher Handling attribute makes the ship turn faster.

2.4. Hoops

Travelling through a hoop triggers an event. Events include granting a timer bonus in Time Trial Mode and updating the Next Hoop indicator.

One hoop will be designated the Start/Finish hoop, indicating the beginning and end of the race course. When a race begins the racers start in front of this hoop. Travelling through the Start/Finish hoop will trigger events such as updating the Lap Count and ending the race. All hoops will be slightly conical to help indicate the direction the ship should travel in. Figure 8 shows some of the initial concepts for the hoop.

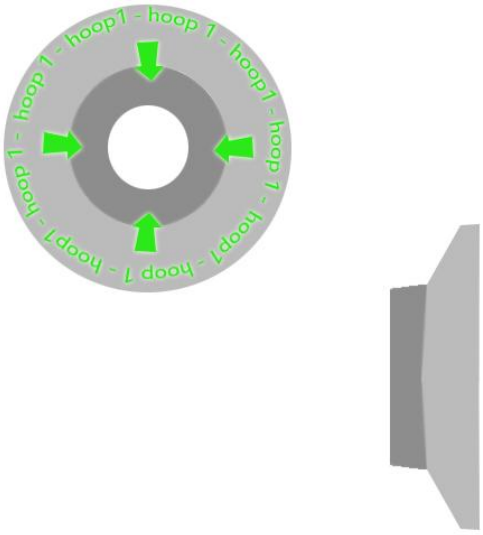

Figure 8 - Hoop Concept

2.5. Obstacles

There are three major distinctions of obstacles: those that have the Obstacle property that damages the ship upon collision, gravity obstacles that alter the ship's trajectory, and the Pirate obstacle, which has a unique behavior.

2.5.1. Asteroids

Asteroid obstacles are scattered throughout the race course. The player's ship receives a small portion of damage if it is hit by an asteroid. The original concept art for the asteroid obstacle is shown in Figure 9.

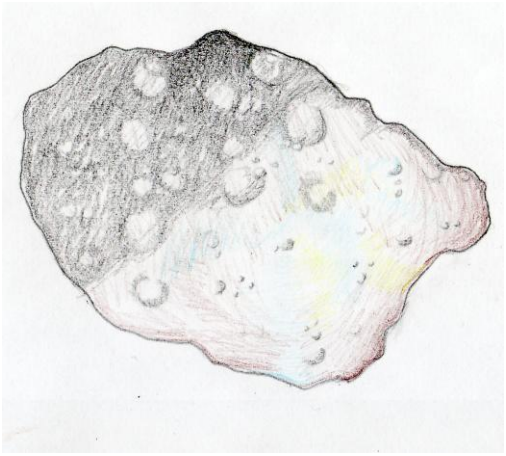

Figure 9 - Asteroid Concept

2.5.2. Asteroid with hoop

This asteroid is larger than the others, with a hole big enough for the ship to travel through. Within this hole, there is a hoop the player must fly through. Figure 10 shows an example of the original concept.

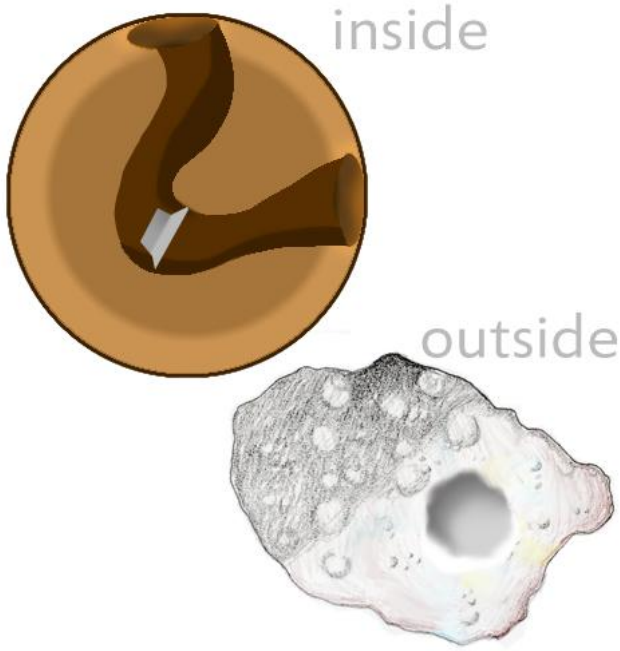

Figure 10 - Asteroid with Hoop Concept

2.5.3. Black hole

The black hole, shown in Figure 11, is an obstacle with a gravitational field. The player is forced to avoid the black hole's gravitational field, or fight the gravitational pull and risk being sucked in and having the player's ship destroyed. There is an audible cue to let the player know that the ship is being pulled towards the black hole once the player gets in range.

Figure 11 - Black Hole Concept

2.5.4. Planet

The planet, pictured in Figure 12, is a large obstacle with a gravitational pull, so players are forced to fight the gravitational pull, or travel far enough away from the planet as to not to be affected. If the player crashes into the planet, the ship takes damage.

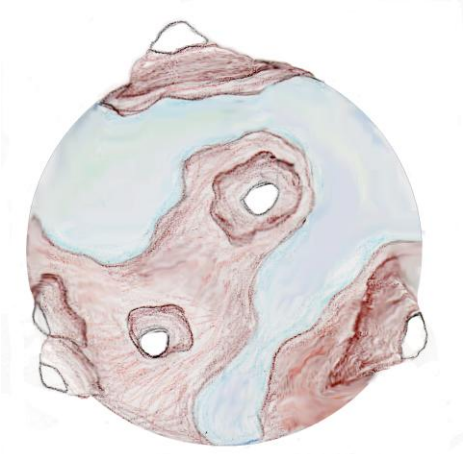

Figure 12 - Planet Concept

2.5.5. Plasma cloud

There are patches of plasma clouds that the player must to avoid while flying, or risk a reduced visibility for a period of time. These plasma clouds are opaque and are scattered in a specific area on the map. The concept art for the plasma cloud is shown below in Figure 13.

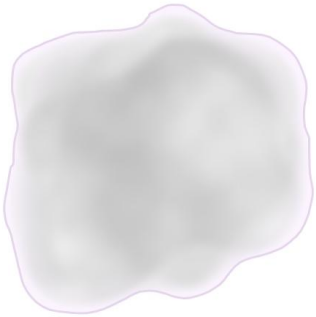

Figure 13 - Plasma Cloud Concept

2.5.6. Solar flares

The sun obstacle emits solar flares, which the player is forced to dodge. If a player gets hit by a solar flare, the ship receives some damage. Figure 14 shows the original design of the solar flare.

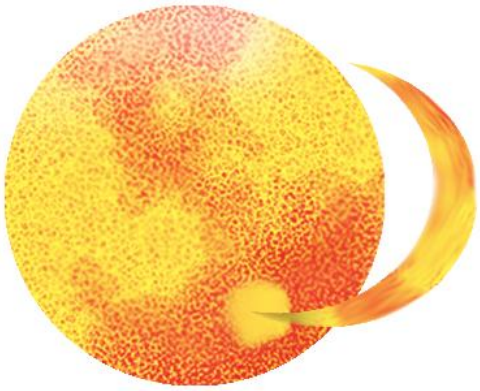

Figure 14 - Sun and Solar Flare Concept

2.5.7. Space pirates

The pirate ship turns and fires projectiles at the player's ship when the player gets close enough. This forces the player to dodge laser shots while continuing around the track. The shots damage the ship if hit.

2.5.8. Man-made doors

The man-made doors are a series of obstacles that move along a given path. Hidden among these doors is a hoop, forcing the player to go through the doors rather than around the obstacle.

2.6. HUD

The Heads-Up Display (HUD) displays vital information to the player. This information is updated upon receiving events from other areas of the program.

The Throttle Indicator (Figure 15, Label 1) shows how fast the ship is traveling based on a percentage of its maximum speed. It is triggered by the Accelerate and Decelerate buttons on the keyboard as well as when a player collides with an obstacle. As the player reaches the next level of speed, the Throttle Indicator updates to fill another bar on the indicator.

The Next Hoop Indicator (Figure 15, Label 2) points toward the next Hoop along the course that the player must travel to. It is triggered by travelling through Hoops. As the player flies around, the indicator arrow orients itself to point towards the next hoop in the hoop array.

The Durability Meter (Figure 15, Label 3) shows how much damage the ship has taken. It is triggered by collision events from obstacles.

The Skill Meter (Figure 15, Label 4) indicates the time until the player has to wait before they may use the Skill again, if one is equipped. The meter drains when a Skill is used, and it slowly regenerates as the race continues.

The Timer (Figure 15, Label 5) is shown during Time Trial and Lap Count modes. In Time Trial Mode, it shows how much time the player has left to travel through the next Hoop on the course. More time is added each time the player travels through a hoop. The race ends when time expires. In Lap Count mode, it indicates how much time has elapsed since the start of the race.

The Race Status (Figure 15, Label 6) indicates to the player how many hoops they have completed in a given lap and also how many laps they have completed. The total hoop number changes depending on how many hoops are in the player's chosen course. The total lap number depends on what the player has specified in the map selection screen.

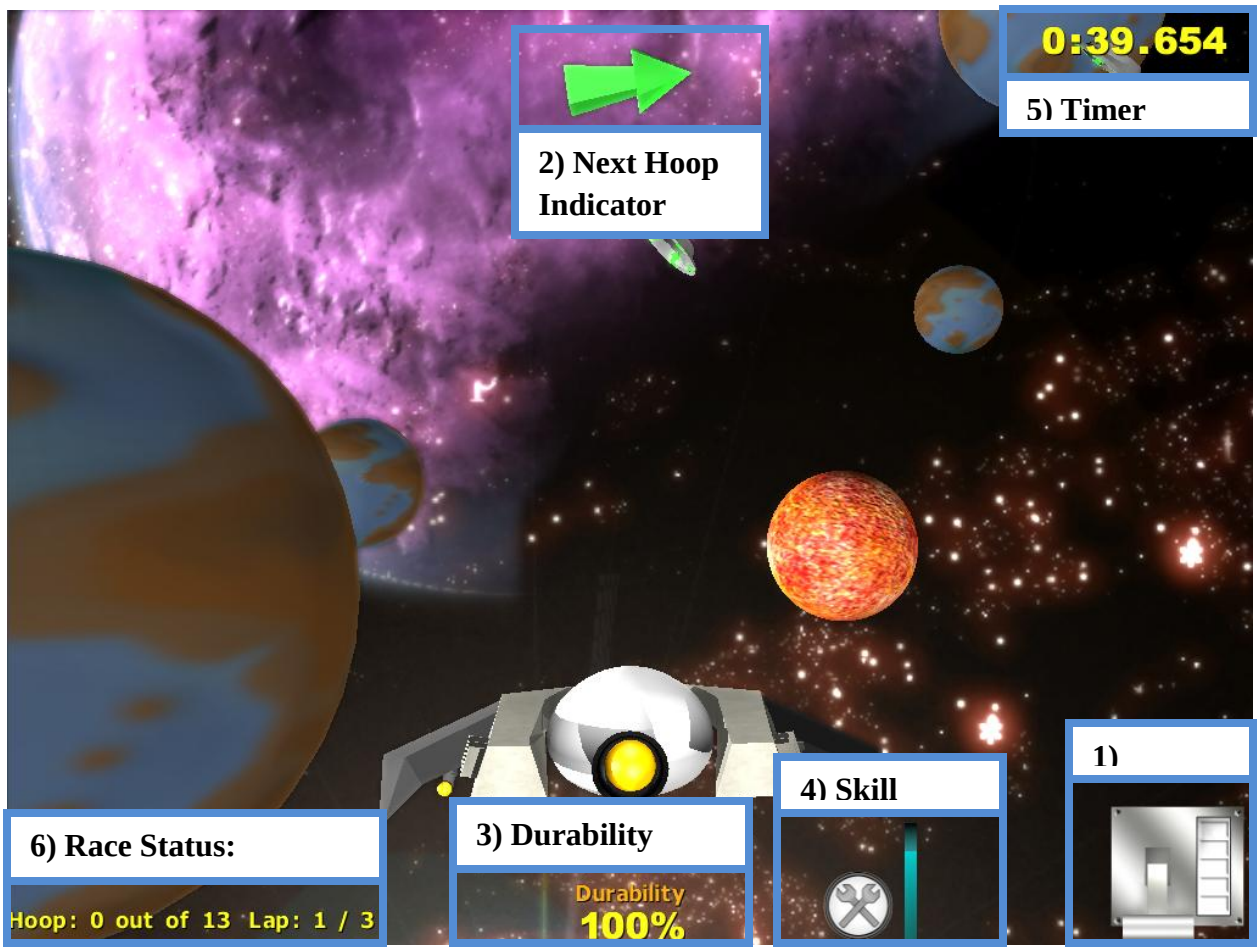

Figure 15- HUD Display Mock Up

3. Art Assets

3.1. Design and Vision

The artistic design calls for a realistic feeling for *Hooping*. Keeping this design in mind, the artistic team aimed for a fast-paced racing game in the three dimensional environment of space, with realistic obstacles, sound effects, and ships.

3.1.1. 2D Art

The 2D Art was created with Adobe Photoshop. This art includes textures for models, as well as layouts for different types of screens. A list of all of the textures that are used in the game can be found in Appendix A Table 6.

3.1.1.1. Menu Screens

The menu screens consist of: the title screen for the game, screens for starting a game and choosing a map, help screens, a credits screen, and the ship-selection screen. Examples of the different menu screens can be seen in Figure 16.

The existing C4 demo included many menu screens that were adapted for *Hooping*, such as the map selection screen. Screens that were custom designed are the help screens and the ship selection menu.

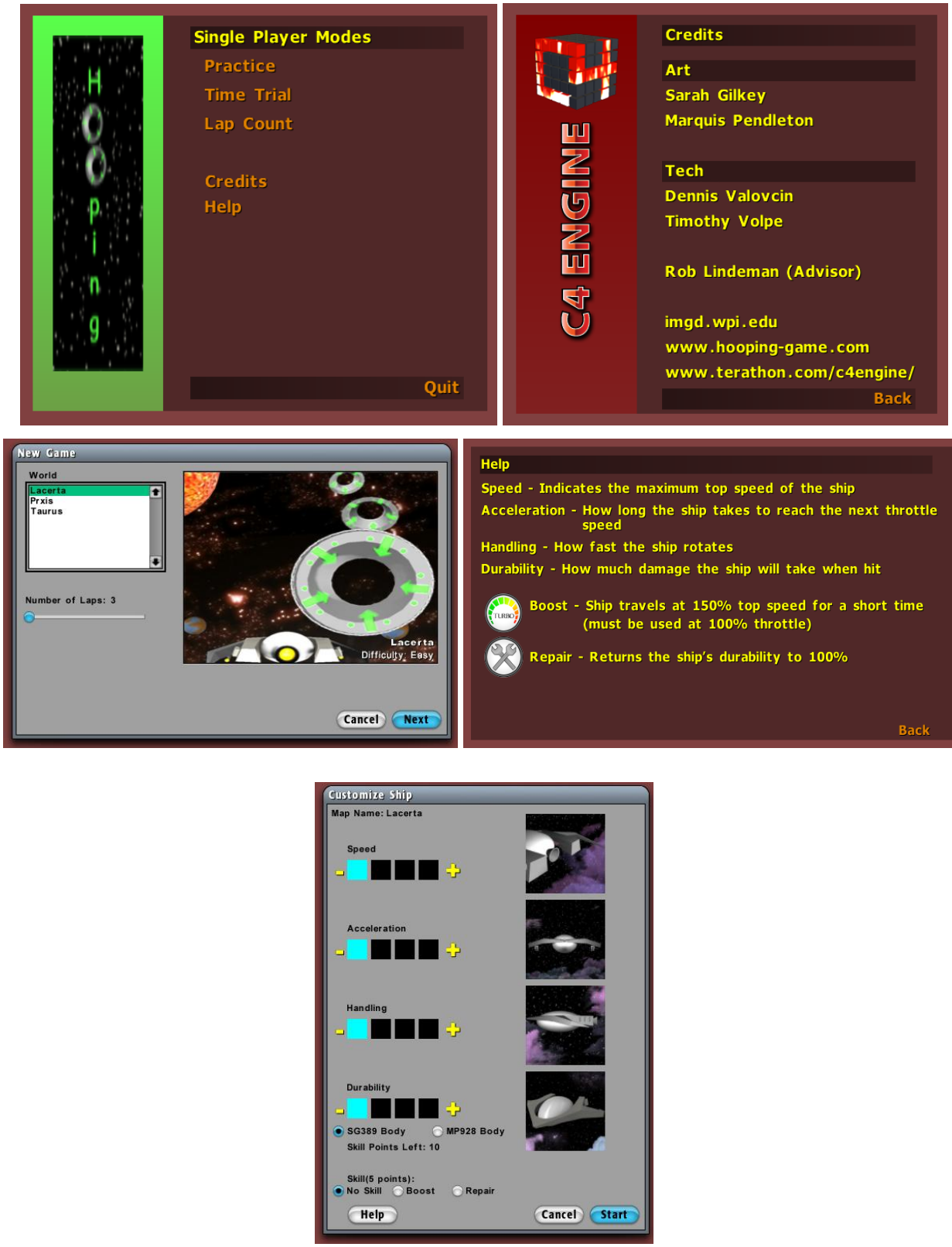

Figure 16 - Menu screens

3.1.1.2. Skybox

The skybox is implemented through a cube shape, with six separate images for each of the six sides of the cube. These textures display an astronomic theme for the maps, whether that is with stars, planets, or other various astronomical objects. This skybox also contains several significant images, of a unique planet for example, to give the players a sense of direction and a way to orient themselves when flying.

There are also several large obstacles that are used in the game like planets and suns, which are enlarged and spread around the course, to enhance the sense of direction for the player, as well as adding realism to the courses.

3.1.2. Audio Design

Audio design was composed using Audacity and FruityLoops. We created all sounds as original content for the game. C4 only supports wav files for music; therefore the game consists of only wav files for all of the audio needed. A list of the audio assets being created for the game can be found in Appendix A, Table 8.

3.1.2.1. Game Music

All of the music in *Hooping* was designed and implemented as original scores to enhance the fast-paced nature of our racing game. The music was designed using up-tempo beats through brass, woodwinds, and percussion instrumental sounds. Descriptions of the different music and sound effects used in the game can be found in Table 1.

Table 1 - Audio Descriptions

Name	*Description*
Menu Screen Music	The menu screen music was designed and implemented as an ambient tune to create a welcoming and inviting feel for players while also setting the mood for the game they are about to begin.
Race Course Music	This ambient music was written to be more driving and give the player a feeling of tension and excitement.

Engine Rumble	The engine rumble is an ambient noise to create a more immersive feel. This sound effect is programmed such that it will fade and grow depending on the acceleration of the ship.
Hoop Ding	The hoop ding is used for when the player passes through the hoop, to signify they successfully completed the correct hoop in the hoop list.
Pirate Gun Zap	In the space pirate obstacle (section 2.5.7), the pirates shoot lasers at the player. These lasers make a zap sound as they are fired.
Solar Flare	The solar flare sound is used by the solar flare obstacle (section 2.5.6) for when the flares hurdle out of the sun, to warn the player of their approach.
Black Hole	The black hole sound is used by the black hole obstacle (section 2.5.3) to help signify that the player is close enough to the obstacle to be affected by its gravitational pull.
Explosion	An explosion sound effect was created for when the ship's durability reaches 0%. Without the sound, it was confusing to the player when their ship died and they respawned or stopped moving. The explosion sound effect gave more clarity to the player when their ship crashed and respawned.

3.2. Implementation

In terms of the artwork, although we were not able to complete everything as it was designed for various reasons, we feel we achieved sufficient success with what was implemented in game. This section discusses the parts of the game that could be implemented, and what was changed from the original design, if anything.

3.2.1. Ships

The two ship bodies were designed to be very unique from each other, so that the player is able to make their ship look unique. With each ship, there are four different textures that change depending on the Durability attribute level selected. In Figure 17, there are images of each of the two ship bodies available in the game.

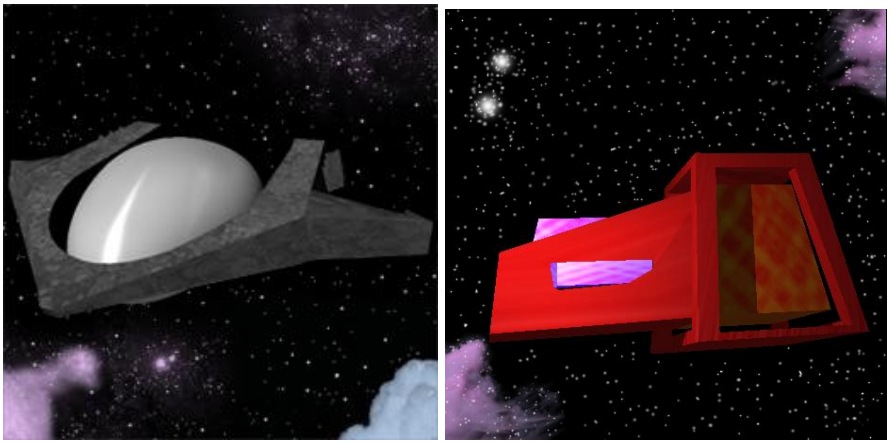

Figure 17 - Ship Body 1 & 2

3.2.1.1. Handling

Handling is shown through the boosters on the ship's wings. There are two boosters created, one large and one small. On the wings, two nodes are attached to the boosters. In code, depending on the level, it was determined if there was supposed to be a booster on the node, and if so, whether it should be a large or small booster. Images of the different booster combinations can be seen in Figure 18. The combinations, from left to right, are: level one with one small booster, level two with one large booster, level three with two small boosters, and level four with two large boosters.

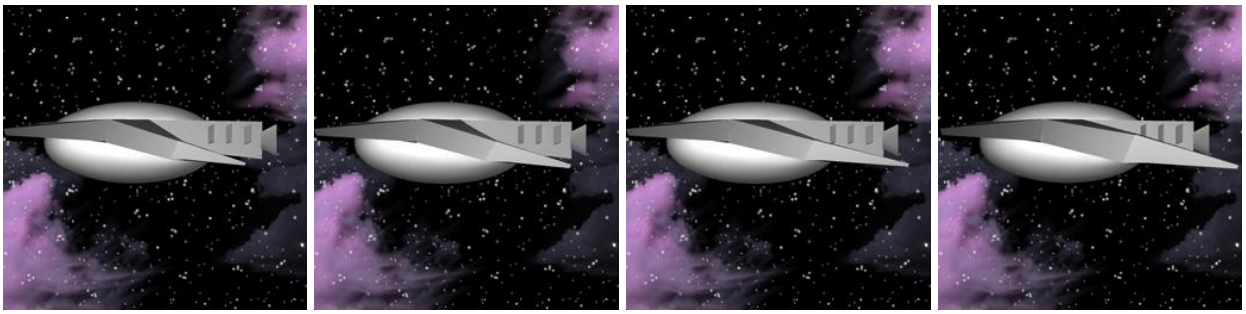

Figure 18 - Handling Levels

3.2.1.2. Speed

Speed is shown through the engines on the back of the ship. The model of the engine becomes larger and more powerful looking with each increase of level. As shown in Figure 19, the speed attribute level increases from left to right, with the left most image being speed level one to the right most image being speed level four.

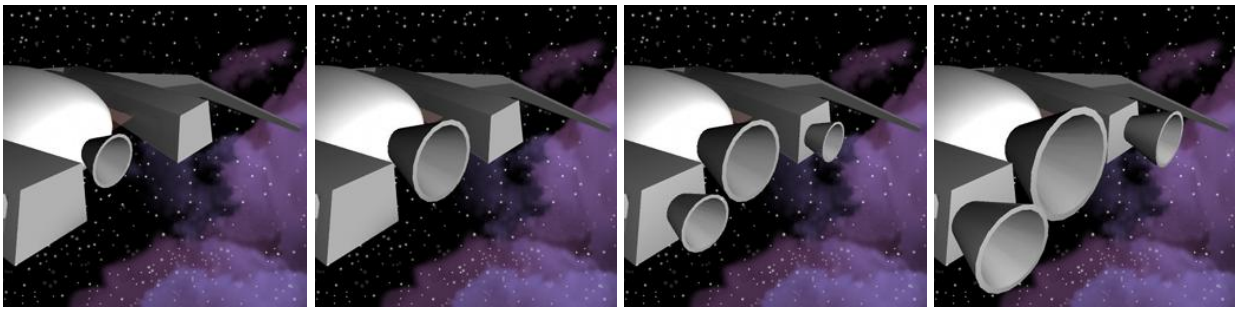

Figure 19 - Speed Levels

3.2.1.3. Acceleration

The wings remain the same no matter which ship body the player chooses. These wings can be seen in Figure 20, with the furthest left image being the lowest level of acceleration and increasing until the highest acceleration on the right.

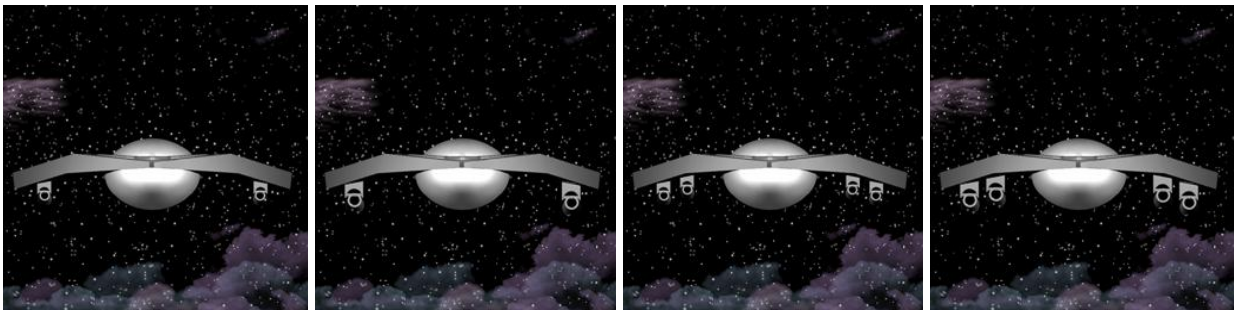

Figure 20 – Acceleration Levels

3.2.1.4. Durability

Durability is shown by the texture on the ship mesh. Each of the bodies that were created was designed with four different textures for the different levels of durability. These textures are designed to appear as though level one was the weakest looking, increasing until the fourth level looking the most durable.

Because the texture is dependent on the ship selected, this is the only attribute that differs depending on the ship selected. Image of the different durability levels can be seen in Figure 21 and Figure 22. These images go from the first ship having durability level one on the left and increasing until durability level four on the right.

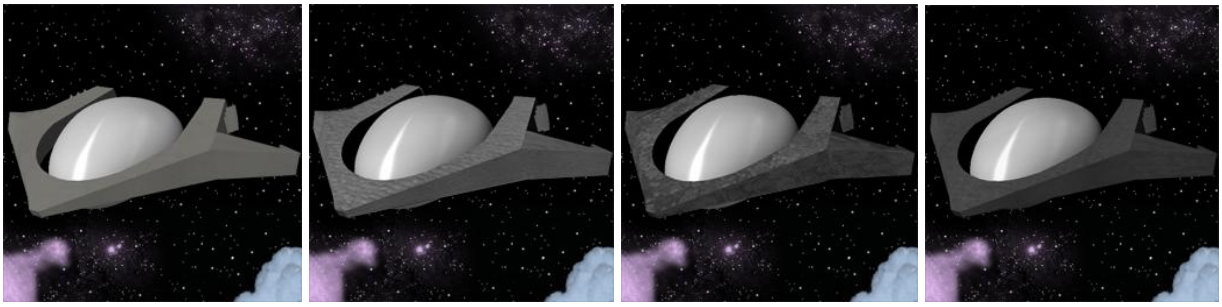

Figure 21 - Durability Levels for Ship 1: SG389

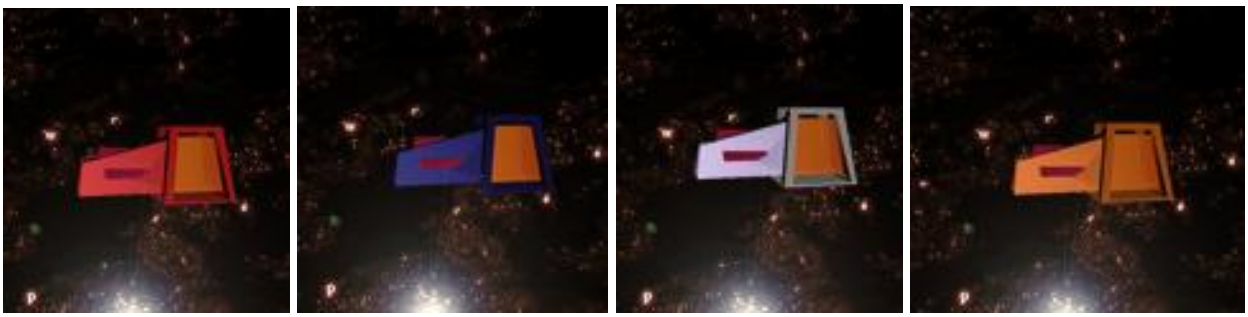

Figure 22 - Durability Levels for Ship 2: MP928

3.2.1.5. Purchasable Skills

Two images, as shown in Figure 23, were created to appear next to the skill bar, to show which skill the player had chosen. The Boost skill (shown on the left) appears as a full speedometer. The Repair skill (shown on the right) appears as two wrenches crossed, which is a common symbol of repair.

Figure 23 - Skill Icons

3.2.2. Hoops

The hoops were designed to be very obvious which direction the player was supposed to fly through. While this is still the case, it was decided that the players would be allowed to fly

through the hoops in “the wrong direction” because it is difficult to fly around the hoop to go through correctly when one is missed during a race. Images of the front, side, and back of the hoop are shown in Figure 24.

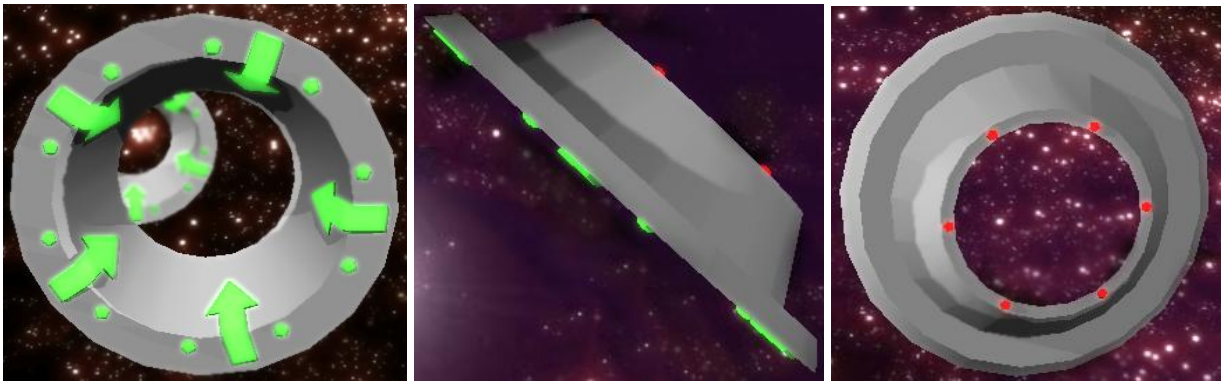

Figure 24 - Hoop

3.2.3. Obstacles

Most of the obstacles were able to be implemented in *Hooping*. A few obstacles were cut because they would have been too complicated to execute within the span of the game. However, we feel we achieved success with the obstacles that were able to be implemented in the game.

3.2.3.1. Asteroid and Asteroid with Hoop

The asteroid was implemented in game as a large asteroid mesh with a hole cut through the middle. A texture was also created for the asteroid to give it a rocky texture. A screen shot of the asteroid as it is found in game is shown in the right image in Figure 25. The left image in Figure 25 is a screen shot of the player flying through the hold in the asteroid.

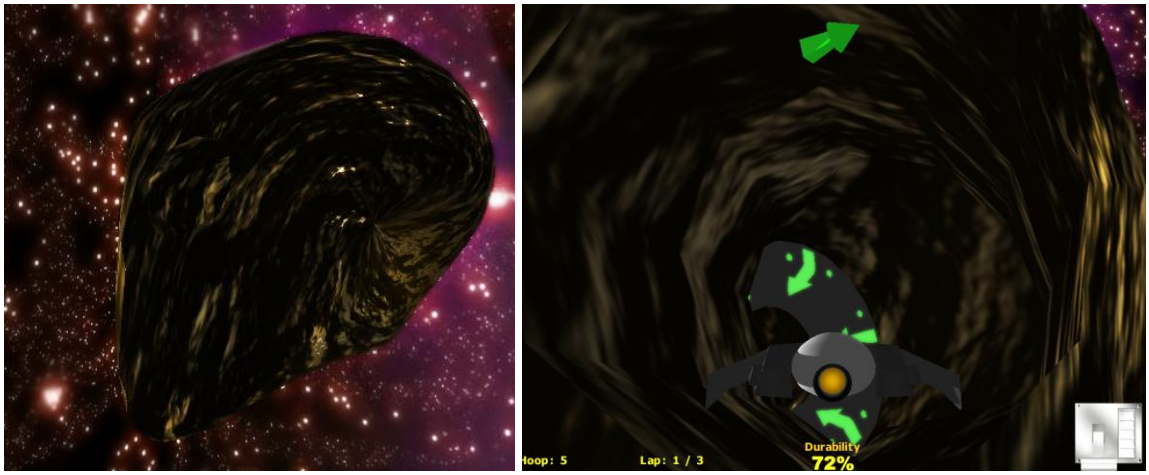

Figure 25 - Asteroid with Hole

3.2.3.2. Black hole

The black hole was modeled as three separate conical meshes, each with its own unique texture. This was done so that each mesh could rotate independently, creating a more realistic feel. The textures for each of the meshes have different colors and transparency channels. A screen shot of the fully implemented black hole obstacle can be found in Figure 26.

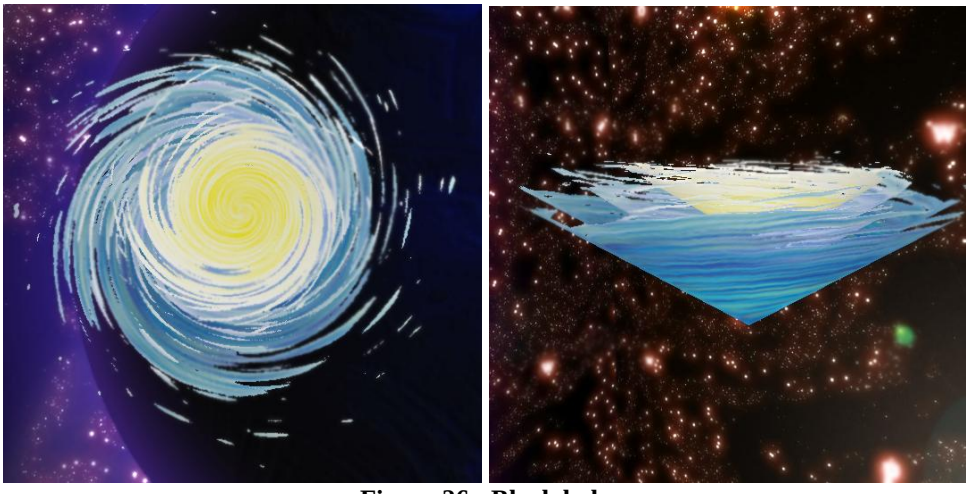

Figure 26 - Black hole

3.2.3.3. Planet

The planet is implemented as a large spherical mesh. This mesh includes a color map as well as a height map, which gives the planet some more interesting texture and makes the continents appear to stick out from the ocean. The texture also rotates slowly around the mesh, giving the feeling of a planet rotating along its axis. See Figure 27 for an image of the planet as it appears in *Hooping*.

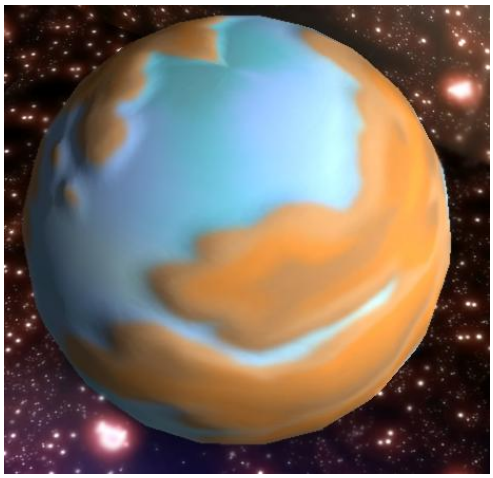

Figure 27 - Planet

3.2.3.4. Plasma cloud

The plasma cloud was implemented as a sculpted geometry. The cloud is not able to be collided with, but it completely blocks the player's view, as seen in the right image in Figure 28. The solid texture serves the purpose of inhibiting the player's view. The HUD arrow still points in the correct direction, but flying without being able to see the course is confusing and intimidating to the player which was the goal for the obstacle.

The Taurus course is designed so that the player must fly through a plasma cloud as soon as they pass through one of the hoops, forcing the player's visibility to be impaired at least once per lap.

Figure 28 - Plasma Cloud

3.2.3.5. Sun and Solar flares

The sun is a spherical mesh similar to the planet. The texture is also moving, like the planet, but in a different direction and speed. The sun model also has a height map applied to it, similar to the planet, to give the sun a bumpy, interesting surface.

The solar flare is a tapered cylinder, which has been deformed into a semi-circle. Images of the sun and solar flare as seen in game can be found in Figure 29. The texture for the solar flare also moves, but very fast, to simulate a fiery texture. In addition to the texture moving, the model itself also rotates around its origin, challenging the player further to avoid this obstacle.

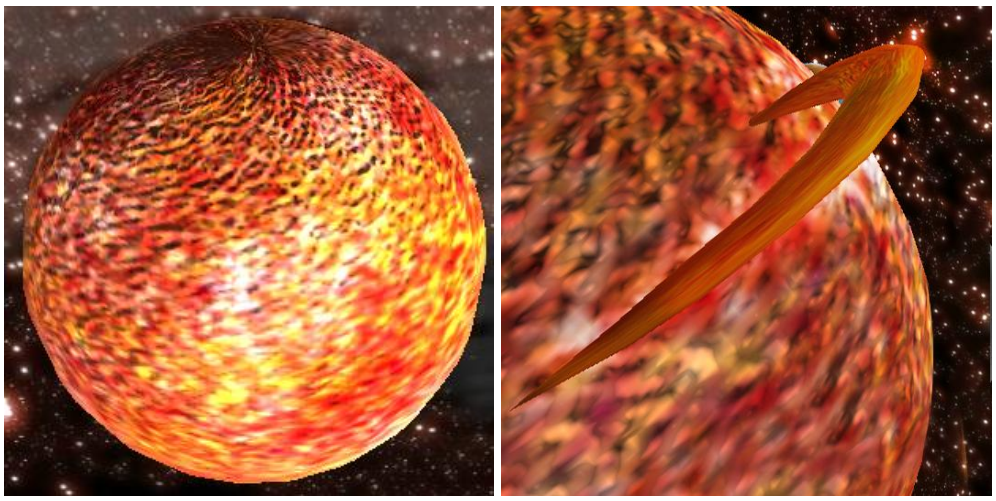

Figure 29 - Sun & Solar Flare

3.2.3.6. Space pirates

The pirate ship obstacle is a ship mesh with cannons on the front to shoot at the player. This ship also re-orients itself to point towards the player's ship when the player flies within a specific range. This ship can be seen in Figure 30.

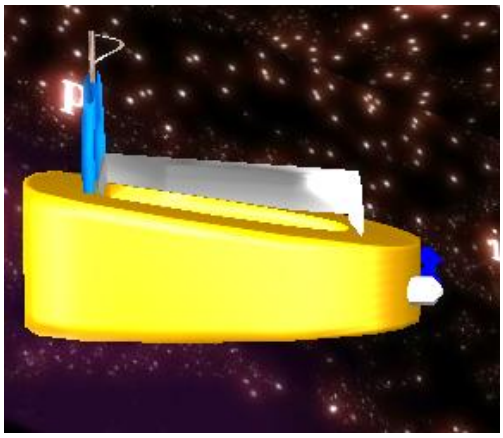

Figure 30 - Pirate Ship

3.2.3.7. Man-made doors

The doors are large cubic meshes, as seen in Figure 31, so the obstacle is much more difficult for the player to maneuver around. The doors move back and forth along a pre-defined path. This obstacle is situated so that if the player flies in between the doors, they may be trapped in the closing doors.

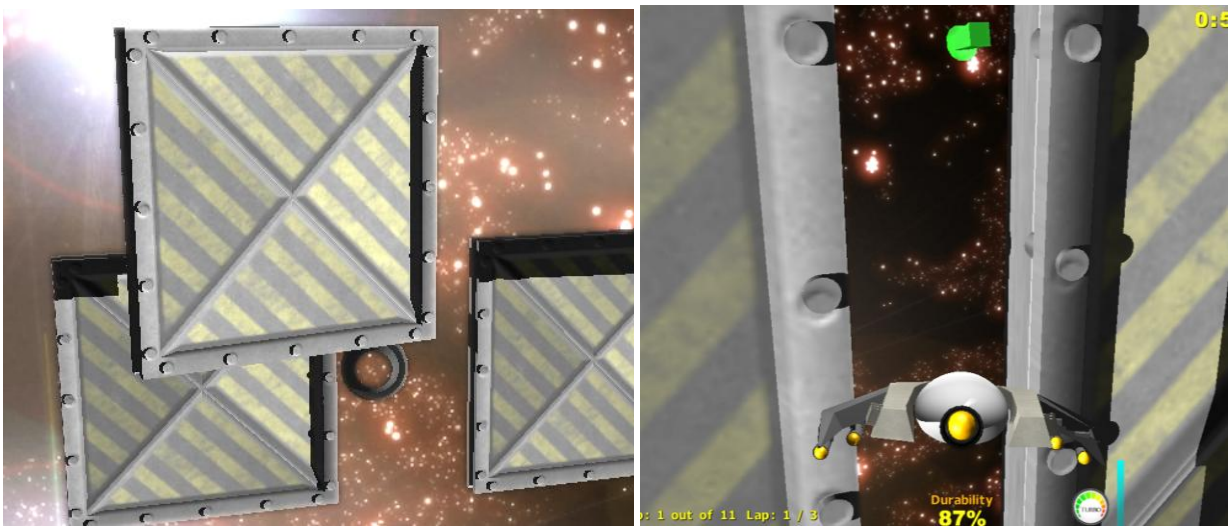

Figure 31 – Man-made Doors

4. Functional Design

For the architecture for *Hooping*, we used Microsoft Visual Studio 2008 to write the program in C++. We utilized the C4 game engine by Terathon Software.

There were many factors in our decision to use C4 for the game engine. One of the main reasons was the flexibility in adapting the existing code to fit our basic needs. C4 comes with a first-person shooter demonstration which provides a lot of the basic functionality we were looking for, including initial setup, camera controls, and sound. Since our game takes place in deep space, we wanted an engine that would grant us better control over physics, especially gravity, and the C4 engine provides this.

Another reason to use C4 is the robust graphics engine, and the built-in effects that it supports, such as particle effects, fog, fire, motion blur and advanced lighting and shading. When it comes to graphics optimization, we have the option of using the C4 portal system to save on rendering cycles. In the end many of the built in effects that drew us to the C4 engine were not used. The effects like motion blur did not work as we hoped and instead just made the ship look blurry. The fire effect that we intended to use for the engines was more a campfire than a jet engine which also turned out to not be useable.

We feel C4's use of a hierarchical scene-graph was useful when it came to ship design, as we planned for our ship to be made up of modular components. The creation of pre-designed race courses was relatively simple thanks to the built-in World Editor.

4.1. Ships

In order to compensate for the fact that the ship should not react to gravity as it is utilized in the C4 engine by default, the ShipController is based on the ProjectileController. The ship reads the mouse movement and applies a rotation to the azimuth and altitude of the model for each frame. Collision detection occurs, and then the ship moves forward according to the camera's direction and speed. When the player presses the Accelerate or Decelerate button, the ship sets a target speed (based on the throttle set), then sets the speed each frame in increments based on the Acceleration attribute.

C4 uses swept volume collision detection that is continuous. For example, a ship is at a point P and the ship's location for the next frame is at point Q. Each frame the ship would make the calculations to determine the point Q. If there are no obstacles in the way, the ship will continue on as normal. Figure 32 shows an example of a normal movement update when no collision takes place.

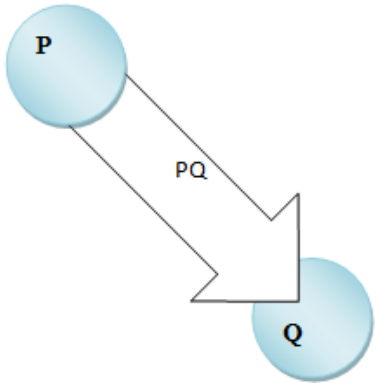

Figure 32 - Movement: No collision

After calculating what point Q would be, a check is made to see if the ship would collide with any obstacles on the way to this position. If the ship collides with an obstacle before reaching this point, the collision is handled accordingly. For example, if the ship were to collide with a wall, the collision handling would stop the ship right in front of the wall and the player would have to navigate away from the obstacle. In Figure 33 it shows an example of the a collision being detected and handled.

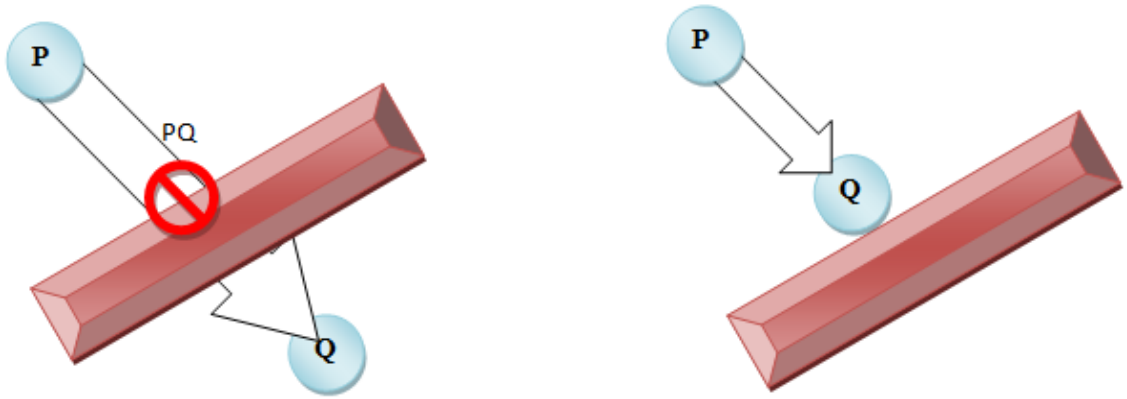

Figure 33 - Movement: Collision

This type of collision was important for a game like *Hooping* where speed would play a factor. By using swept volume collision, it ensures that collisions would not be missed simply because the ship was moving too fast or the collision detection did not happen often enough. For example, under a different collision scheme, if the ship travels too fast, it may reach point Q in one frame and completely bypass the collision object, thereby passing through the object unharmed.

Should the ship collide with an object, another function is called so the ship's speed is drastically altered so as to not pass through the object's geometry. If said object is assigned the Obstacle property, this ship will take damage depending on how much throttle is applied to the ship at the time. Should the ship's durability rating drop below 100%, the ship comes to a complete stop and the throttle is reset to 0%.

If a Skill is assigned to the ship, the controller registers the Activate Skill button press. If the Skill is set to Boost, the ship's speed is increased by 150%, but only if the ship is set to 100% throttle. The ship's durability is reset to 100% at any time if the Repair Skill is chosen. In both cases, once the skill is used, the skill meter is drained, and there is an internal timer that recharges the skill meter over a period of time.

4.2. Obstacles

Almost all of the obstacles have an Obstacle Property attached to them. Geometry nodes are assigned this property in the World Editor. When the ship controller detects the collision, it can get the properties of the object it collided with. If the ship collides with a node with the Obstacle property it handles the damage accordingly. The other benefit of using a property is that it can be assigned independently of object controllers. A node could have a controller that causes the obstacle to move or spin and the obstacle property would still be responsible for making sure the ship took damage.

4.2.1. Black Hole and Planet (Gravity Obstacles)

These obstacles are defined in the World Editor along with a sphere-shaped trigger area that surrounds them. These trigger areas are supplied with the origin point of the obstacle and a strength value. When the ship enters this trigger, it calls a function that notifies the ship to alter its velocity.

When this function is called, a unit vector is calculated between the ship's current location and the obstacle's origin point. This vector is scaled up by the obstacle's strength, and added to the ship's current velocity. This, in effect, achieves a simulated gravitational pull.

4.2.2. Plasma Cloud

The Plasma Cloud is little more than a geometry added in the World Editor. When the ship passes into it, the geometry blocks the player's view in an attempt to disorient the player.

4.2.3. Pirate Ship

When the ship passes within a specified range of the Pirate Ship obstacle, the Pirate rotates to continually follow the player ship. Additionally, the Pirate periodically launches a Fireball (borrowed from the built-in C4 demo) at the player which, upon collision, will cause some damage to the ship.

4.2.4. Man-made Doors

The Man-made doors are low level obstacles. The doors needed to have the ability to move along any axis for any distance. Furthermore we wanted to have the option for the doors to repeat their movement, patrol back and forth, or travel a certain distance and jump back to the beginning, all at varying speeds. The most effective way to do this was by creating a new controller that could be assigned to nodes in the World Editor. After assigning the controller, a movement speed, travel distance, travel axis and repeat option could be set.

4.3. Game Mechanics

For all of the game modes, the game needs to keep an ordered list of the hoops. As the game loads a level, the LoadWorld function iterates over every node in the game. If the node has the hoop controller assigned, it gets added to the overall hoop list with the hoop number being the array index. In that way, the hoops are stored in the array in the proper order.

Once in a race, the player will be flying through hoops to complete the laps. The hoops are made up of two parts, the visual hoop geometry and an invisible hoop trigger. Each of these two pieces served individual purposes. The hoop geometry makes use of a hoop controller. When assigning the hoop controller in the World Editor, a hoop number is assigned that is used when

the hoop list is created. When the hoop trigger is activated, it does a check to make sure that it is the correct hoop the player should go through. If it is the correct hoop, a sound will play as an audio award for making it through the hoop. Along with playing a sound, the game HUD is also updated, displaying the last hoop the player went through, and if the player just finished a lap, the lap count is also updated. Finally, if the Time Trial mode was selected, the game adds an additional ten seconds to the timer.

The game engine uses a function called ApplicationTask, which is called every frame. This function is used to update the in-game race timer. When the race starts, the system time is recorded, and the game keeps track of the amount of time in milliseconds that have elapsed since then. This value is passed to the DisplayInterface, which translates it into an appropriate value of minutes and seconds and writes it to the screen. This function also makes note of when a certain amount of time has passed for game-end conditions.

4.3.1. Display Final Results

When a race ends, the final results are shown. For Time Trial mode, these results include whether or not the player won the race, how many laps were completed, and how many hoops on the current lap were triggered. In Lap Count mode, the results shown include the number of laps the player completed and the total elapsed time.

5. Project Timeline

This timeline is our proposed schedule as we had planned during the design phase of the project, including things we eventually were forced to remove from the game implementation for various reasons.

5.1. Design Schedule

Table 2 details the initial design schedule for *Hooping*.

Table 2 - Design Schedule

Date Due	Task Due
August 29, 2008	Design Begins
September 8, 2008	Project proposal document and presentation
September 23, 2008	Initial design document and presentation
October 7, 2008	Critical design document and presentation

5.2. Implementation Schedule

The implementation schedule, found in Table 3, lists the technical execution schedule for the game.

Table 3 - Implementation Schedule

Date Due	Task Due
November 2, 2008	Code preparations and running an empty world
November 9, 2008	Input controls complete, ship object coded
November 16, 2008	Hoop object coded, HUD display created in code
November 23, 2008	Game modes programmed, obstacle object coded
November 30, 2008	Completing single player modes
December 7, 2008	Multiplayer menus and messaging

December 14, 2008	Completing multiplayer modes
December 18, 2008	End of B Term; Fully implemented game

5.3. Assets Schedule

The schedule of artistic assets is listed in Table 4.

Table 4 - Assets Schedule

Date Due	Task Due
November 9, 2008	Ship models created, hoop model created
November 23, 2008	Obstacle creation and upgrade model complete
November 30, 2008	Completing single player modes
December 7, 2008	Sound effect creation and menu art
December 14, 2008	Music creation for courses and introduction
December 18, 2008	End of B Term; Fully implemented game

5.4. Testing Schedule

The testing schedule, in Table 5, lists the initial testing plan for the game. An example of our testing plan can be found in Appendix B.

Table 5 - Testing Schedule

Date Due	Task Due
February 2, 2009	Content freeze, testing begins
February 9, 2009	Menu and game mode testing
February 16, 2009	Course and ship attribute testing
February 23, 2009	Obstacle testing
March 6, 2009	End of C Term

6. Project Results

6.1. Methodology

We approached this project with an agile implementation process. Every week we presented a demonstration of what we had accomplished. This helped our team to re-evaluate our progress each week and discuss the tasks we felt were most important to accomplish in the coming week. We were also forced to review our design in the middle of implementation to discuss the feasibility of our original design's ability to be executed in the time we had remaining in the project.

6.2. Successes

Though time constraints forced the scaling of *Hooping* to remove multiplayer mode, we were able to complete a fully functional game. The three single player modes: Practice, Time Trial, and Lap Count, are all completely implemented and working as designed. The obstacles are all completely operating in code.

The *Hooping* team also was able to successfully go through the full development cycle, from a fully thought out design document, through the entire implementation process, and through an internal testing and bug fixing stage. We as a group have gained invaluable knowledge about this process.

6.3. Problems

The biggest problem our group had was with the time constraints a three term MQP put on the game. The *Hooping* team had very large plans for the game, but difficulties early in the implementation process, especially when attempting to create the ship controller, lead to a setback in our schedule and ultimately caused a reworking of the timeline.

The C4 Engine's lack of documentation also caused problems with implementation. This engine is originally designed as a first person engine, where objects must follow the constraints of physics and gravity. *Hooping*, however, is designed as a game in which the player must move in all three dimensions without experiencing the effects of gravity on the vehicle. With some outside help, we were able to finally solve the problem.

The pipeline for getting models and textures into the C4 engine took the artistic team several weeks to master, which also caused problems for getting models into the game. This problem lead to many of the models previously constructed to be rebuilt to fix problems with shading, normals, and textures that arose from importing.

6.4. Analysis

After reviewing the end result of our project, there are several things we would have done differently had we known then what we do now. These are:

1. More Technical designing and planning. The technical team discovered during the implementation stage of our game that they had not planned the functionality of the game out as well as needed. They assumed they would be able to use more of the code that comes with the C4 engine, however, this code proved to not be useful for what they needed with *Hooping*. Having more technical planning would have allowed the technical team to progress more rapidly, which would have allowed more time for new features or more testing.

2. Working out the problems with the art pipeline sooner. The art team had a difficult time getting models created in Maya through the process of exporting from Maya and importing into the C4 engine without common problems like incomplete geometry or incorrect textures. If this process had been worked out sooner, the art team would have been able to get more models into the game much earlier in the process.

3. Look into C4 earlier, discover what was possible earlier, and prototype earlier during design. Throughout the design process, we did not look into the capabilities that the C4 engine had. This lead to designing features which were impossible to implement in the scope of this project. If our group had looked into C4 more during the design process, we would have been able to design a game that would have been much easier to implement.

4. Team management, no leader, and more team meetings. From the beginning, our group had no clear leader. Without leadership, there was no one person that kept the group

accountable for its actions. This caused problems with all group members staying on task and getting their work completed in the time scheduled.

7. Conclusion

Hooping is a racing game where players are challenged to fly through hoops along a given course. There are a variety of obstacles between each hoop to test the player's flying skills. *Hooping* offers players three single player modes of play: practice, time trial, and lap count. The game offers players the chance to upgrade desired skills through the ship selection menu, so that each player can have advanced control over how their ship flies.

Overall the project was a valuable learning experience. We were able to go through the entire design cycle from start to finish. We experienced the issues that come with design, and in our case, over-designing, and how a game can adapt and evolve over the course of the implementation cycle and testing cycle to a finished product.

While leadership and time management were hindering factors throughout the course of the project, the game really started to come together toward the end. In our final term of the project, we were able to step up the development and were able to refocus, allowing us to complete many major tasks that had been unfinished. While we had high hopes for the game, the team as a whole is satisfied with the outcome.

Appendix A. Art Asset Listing

Table 6 - Texture Asset Listing

Name	*Description*	*Source*	*Completion*
Meteor	512x512, for Meteor, Comet, and Asteroid obstacle	Photography	
Space Pirate Laser	128x128,For Space Pirate obstacle	Original Creation	
Sky Box	6 images in total, 512x512, for use as background images in courses	Original Creation, Photography, Online	
Introduction Screen Art	512x512 for use in introduction page to replace C4 logo	Original Creation	
Ship Selection Images	256x256, for use in ship selection, show variations of ship parts	Original Creation	

Table 7 - Model Asset Listing

Name	*Description*	*Source*	*Completion*
Ship 1 body –upgrade 1	Basic level ship body for stock ship 1 mesh + 0% armor texture	Original creation	
Ship 1 body –upgrade 2	Second upgrade of ship body 1 mesh + 25% armored texture	Original creation	
Ship 1 body –upgrade 3	Third upgrade of ship body 1 mesh+ 50% armored texture	Original creation	
Ship 1 body –upgrade 4	Final upgrade of ship body 1 mesh + 100% armored texture	Original creation	
Ship 2 body –upgrade 1	Basic level ship body for stock ship 2 mesh + 0% armored texture	Original creation	
Ship 2 body –upgrade 2	Second upgrade of ship body 2 mesh + 25% armored texture	Original creation	
Ship 2 body –upgrade	Third upgrade of ship body 2 +	Original creation	

3	50% armored texture		
Ship 2 body –upgrade 4	Final upgrade of ship body 2 + 100% armored texture	Original creation	
Ship 3 body –upgrade 1	Basic level ship body for stock ship 3 mesh + 0% armor texture	Original creation	
Ship 3 body –upgrade 2	Second upgrade of ship body 3 mesh + 25% armored texture	Original creation	
Ship 3 body –upgrade 3	Third upgrade of ship body 3 mesh + 50% armored texture	Original creation	
Ship 3 body –upgrade 4	Final upgrade of ship body 3 mesh + 100% armored texture	Original creation	
Ship 4 body –upgrade 1	Basic level ship body for stock ship 4 mesh + 0% armor texture	Original creation	
Ship 4 body –upgrade 2	Second upgrade of ship body 4 mesh + 25% armored texture	Original creation	
Ship 4 body –upgrade 3	Third upgrade of ship body 4 mesh + 50% armored texture	Original creation	
Ship 4 body –upgrade 4	Final upgrade of ship body 4 mesh + 100% armored texture	Original creation	
Wing - upgrade 1	Basic level wing (base size) mesh + texture	Original Creation	
Wing - upgrade 2	Second upgrade of wing (130% original size) mesh + texture	Original Creation	
Wing - upgrade 3	Third upgrade of wing (170% original size) mesh + texture	Original Creation	
Wing - upgrade 4	Final upgrade of wing (200% original size) mesh + texture	Original Creation	
Engine - upgrade 1	Basic level engine (at base size) mesh + texture	Original Creation	
Engine - upgrade 2	Second upgrade of engine (130% original size) mesh + texture	Original Creation	

Engine - upgrade 3	Third upgrade of engine (170% original size) mesh + texture	Original Creation	
Engine - upgrade 4	Final upgrade of engine (200% original size) mesh + texture	Original Creation	
Booster - upgrade 1	Basic level booster (at base size) mesh + texture	Original Creation	
Booster - upgrade 2	Second upgrade of booster (130% original size) mesh + texture	Original Creation	
Booster - upgrade 3	Third upgrade of booster (170% original size) mesh + texture	Original Creation	
Booster - upgrade 4	Final upgrade of booster (200% original size) mesh + texture	Original Creation	
Engine Flame	Flame used in engines and boosters	Original Creation	
Hoop	mesh + texture	Original Creation	
Asteroid	mesh + texture	Photography	
Black Hole	Spinning animation, mesh + texture	Original Creation	
Planet	Rotating animation, mesh + texture	Original Creation, Photography	
Plasma Cloud	Changing animation, mesh + texture	Original Creation	
Sun	Rotating animation(For Solar Flare obstacle), mesh + texture	Original Creation, Photography	
Solar Flare	mesh + texture	Original Creation, Photography	
Space Pirate Ship	Shooting animation, mesh + texture	Original Creation, Photography	

Man Made doors	mesh + texture	Original Creation, Photography	

Table 8 - Audio Asset Listing

Name	*Short Description*	*Source*	*Completion*
Menu Screen Music	WAV clip, ambient (plays throughout menu system)	Original Creation	
Course Music	WAV clip, ambient (plays throughout course 1)	Original Creation	
Engine Rumble	WAV clip, event (plays when player is in First Person mode)	Original Creation	
Hoop Ding	WAV clip, event (plays when player passes through a hoop correctly)	Original Creation	
Gun Shooting	WAV clip, event (plays whenever a mouse click is detected)	Original Creation	
Solar Flare	WAV clip, event (plays when player is near a Solar Flare obstacle)	Original Creation	
Black Hole	WAV clip, event (plays when player is near a Black Hole obstacle)	Original Creation	
Explosion	WAV clip, event (plays when player's durability reaches 0%)	Original Creation	

Appendix B: Testing Plan

Hooping Testing Plan		
Part to be tested	Individual test items	How to test
ship	all parts lined up	Select all possible combinations of ship, look at body-wing, wing-booster, and body-engine fittings
	durability level effects damage taken	Start from a set point away from obstacle, fly at max acceleration into obstacle at each durability level, time to see how long it takes to go from 100% to 0%
	speed not too fast/slow	Player Feedback
	handling does not turn too fast	Player Feedback
	acceleration does not turn too fast	Player Feedback
	repair skill work as expected	Start race with a set amount of damage, use skill, verify damage is restord
	boost skill work as expected	Use skill at all levels of throttle, verify that ship speed increases properly
	skill bar increases/decreases when used as expected	Use skill, verify bar decreases when used and then begins to increase propertly after skill finishes
obstacles	gravity pulls you towards it	Start from a set point away from gravity obstacle, fly at obstacle, note if vector traveling at changes
	gravity escape velocity w/different speeds	Start from a set point away from gravity obstacle, fly at obstacle, once gravity begins to affect path, turn away from obstacle and max accelerate away, note if can espace gravitational pull
	pirate ship fires correctly at player	Fly on a set path near a pirate ship obstacle, note if ship fires or not
	pirate ship does damage to ship w/all durability levels	Fly on a set path near a pirate ship obstacle, allow pirate to damage ship, note if fired bullet does damage
	sounds work with correct obstacle	Start from a set point away from all obstacles, as fly near each obstacle, note if sound begins
courses	all hoops able to fly through	Start at beginning of each map, fly through each hoop in order, note if trouble flying through any hoop and which one causes problems
	course difficulty fits course layout	Player Feedback
	time trial winnable for each course	Player Feedback
	lap slider works correctly for each course	Go though menu screens, select lap count and time tiral, select laps 3-10 for each map, then enter game and verify that laps required equals laps selected
	hoop arrow points to each next hoop	Start at beginning of each map, fly through each hoop in order, verifying that as soon as hoop is passed through, the next hoop is pointed to