

Claustrosphere

Game Design Document

v2.6 – April 19, 2007



Bats=Death!

Bats = Death!
Joseph Boyd
Caleb Lee
Brandon Riggs

Fall 2005
GAM 400
Prof. Moore

Table of Contents

Revision History.....	5
Introduction.....	6
Game Concept.....	6
High Concept.....	6
Core Game Play.....	6
Theme.....	6
Back Story.....	6
Example of Play.....	8
Market Analysis.....	9
Target Market.....	9
Expected ESRB Rating.....	9
Competitive Analysis – Geometry Wars: Retro Evolved.....	10
Competitive Analysis – Mars Matrix: Hyper Solid Shooting.....	11
Competitive Analysis – Tetris.....	12
Cost Analysis.....	13
Project Overview.....	14
Target Hardware Requirements.....	14
Milestone Schedule.....	14
Team Members.....	14
Game Mechanics.....	16
Game Walkthrough.....	16
The Player.....	17
Character Design.....	17
Movement.....	17
Abilities – State Change.....	17
Abilities – Fire.....	18
Progression – Leveling.....	18
The Sphere (The Playfield).....	20
As Dynamic Playfield.....	20
As Health/"Game Over" Trigger.....	20
As Score Modifier.....	21
The Enemies.....	22
Color Coordination.....	22
Visual Effects.....	22
Tetrahedron (4 Faces).....	22
Cube (6 Faces).....	23
Octahedron (8 Faces).....	23
Dodecahedron (12 Faces).....	23
Icosahedron (20 Faces).....	24
Enemy Fusion.....	24
The Fusion Factor.....	24
Fusion Compatibility.....	24

Icosidodecahedron (32 Faces).....	25
A Final Note on Fusion.....	25
Enemy Compounds.....	25
The Ultimate Enemy.....	26
Sphere.....	26
Enemy Spawning.....	27
Waves.....	27
Spawn Groups.....	28
Enemy Type Selection Process.....	29
Enemy.....	30
Weight.....	30
Probability.....	30
Lookup Values.....	30
Enemy Compound Selection Process.....	30
Pulses (Pick-Up Objects).....	31
Red Pulse.....	31
Blue Pulse.....	31
Power Pulse.....	31
Pulse Spawning.....	31
Scoring.....	32
Base Score.....	32
Sphere Modifier.....	32
Time Modifier.....	32
Putting It All Together.....	33
Displaying the Score.....	33
Special Effects.....	34
The Sphere.....	34
State Change.....	34
Sphere Size Change.....	34
Enemies.....	35
Projectiles.....	35
The Player.....	35
Game Over.....	35
Appendices.....	36
Appendix A – Game Modes.....	36
Normal Mode.....	36
Onslaught.....	36
Survival.....	36
Pacifist.....	37
Appendix B – Interface Details.....	38
Menu – Main.....	38
Menu – Options.....	38
Menu – In Game (Pause).....	38
Menu – High Scores.....	39
Menu – Credits.....	39

Menu – Graphics Options.....	39
Menu – Sound Options.....	39
Menu – Control Options.....	39
Menu – Game Over/Enter Name.....	40
Menu – Game Type Specific High Scores.....	40
Controls – In Game.....	41
Controls – Menus.....	41
In Game – HUD.....	41
Appendix C – Game Flow.....	42
Appendix D – Project Schedule.....	43
Schedule Overview.....	43
Engine Proof Checklist.....	43
First Playable Checklist.....	43
Pre-Alpha Checklist.....	44
Alpha Checklist.....	44
Beta Checklist.....	44
Gold Master Checklist.....	44
Appendix E – Enemy Stats.....	45
Appendix F - Team Bats = Death! Information.....	46
Joseph Boyd.....	46
Project History.....	46
Sign Off.....	46
Caleb Lee.....	47
Project History.....	47
Sign Off.....	47
Brandon Riggs.....	48
Project History.....	48
Sign Off.....	48

Revision History

Ver.	Date	Author(s)	Comment
1.0	09/29/06	Joe Boyd Caleb Lee Brandon Riggs	Initial document construction.
2.0	10/28/06	Caleb Lee	Changed wording in some areas to be more explicit, added the " Scoring " section under " Game Mechanics ." This was a revision to net a few more points for our second turn-in of the GDD.
2.1	12/11/06	Caleb Lee	Added a Revision History. Updated " Controls – In Game " to include roll (for some reason it wasn't there).
2.2	01/23/07	Caleb Lee	Changed " The Sphere " to our new specs of being transparent in a skybox. Updated our " Schedule Overview " per Jen Sward's request. Updated the " Enemy Stats " to be a bit easier at the beginning. Did some minor formatting updates. Consolidated the duplicate Visual Effects sections into " Special Effects ".
2.3	02/01/07	Caleb Lee	Removed all mention of State Changes and their effects, added " The Centroid " to our document to reflect the change in the game design. Added two score modifiers (" Density Modifier " and " Time Without Damage Modifier ") and adjusted the Modifier application explanation in " Putting It All Together ". Added " The Centroid " to the special effects section. Adjusted some grammatically questionable phrasings. Removed the mention of "jumping" when double tapping a strafe key in the " Movement " as that feature was nixed a long time ago but was overlooked in this GDD. Added date column to this table.
2.4	02/06/07	Caleb Lee	Changed the term "Projectiles Per Second" (PPS) to "Shots Per Second" (SPS).
2.5	02/13/07	Caleb Lee	Fixed some wording in " Joseph Boyd ".
2.6	04/19/07	Caleb Lee	Final Revision this school year. Gutted this bad mama jama of all the things that didn't make the final cut. *tear*

Introduction

Game Concept

High Concept

Claustrosphere is a third-person 3-D action-arcade game where the player takes control of a ship to destroy hordes of enemies and rack up as many points as possible. In order to accomplish this objective, the player must collect power-ups to upgrade his weapons and deftly maneuver through the playfield while shooting enemies. Inspired by simple, addictive games such as *Geometry Wars* and *Tetris*, the massive amounts of graphical and sound effects combined with fast-paced and chaotic action will keep the player coming back for more.

Core Game Play

The gameplay will be solid, fast-paced, and chaotic. Using the mouse and keyboard, the player will fly around in full 3-D and continually shoot bullets at enemies. The player also has the ability to control the spread of his fire; in order that the bullet spray may be focused to take down a single target quickly or widened to cover a larger area. The enemies will be numerous and spawn in waves. Occasionally enemies will drop power-ups that the player may pick up to make his gun stronger. As the player's weapon is upgraded, he or she is able to shoot more bullets at a faster rate and widen the cone of fire more.

The playfield itself is a sphere that is constantly shrinking, and the player must collect energy dropped by enemies to keep the sphere from imploding. If the sphere collapses, then it's game over. Keep in mind, however, that the smaller the sphere is, the more points the player scores for defeating enemies. Thus there is an intriguing balance between maximizing the score and leaving enough room to survive. As the game progresses, enemies become harder and more numerous, slowly building tension and keeping the player entranced.

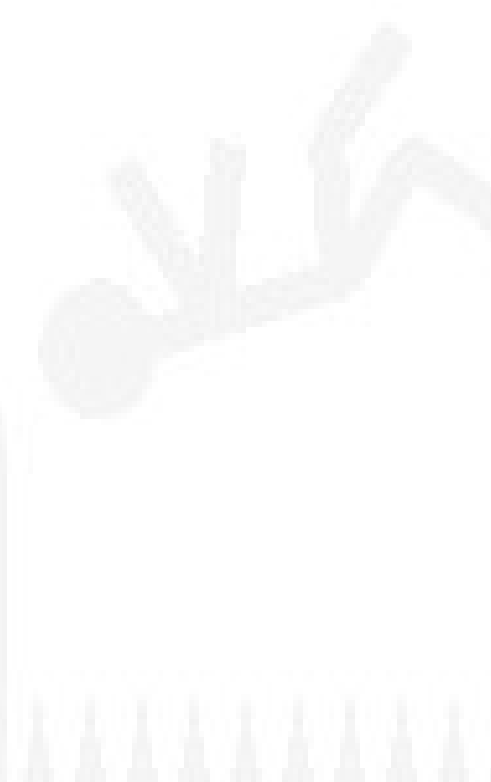
Theme

The primary goal for *Claustrosphere* is to maximize amplification of input, where trivial button presses result in countless visual and aural stimuli. The control scheme will be simple and intuitive enough for anyone to pick up immediately. There will be profuse amounts of graphical effects such as multitudes of moving enemies, particles, and firework-like explosions. Riveting sound effects will also help make the game come alive. All of these characteristics will combine to give the player a huge sense of power and accomplishment from expending very little effort.

Artistically, the game will be very abstract. The playfield itself is a sphere, and all game characters are polyhedrons. There will be a wide variety of colors—covering the entire spectrum of the rainbow—used to represent each type of character.

Back Story

Far along a dimension that we have yet to fathom there exists a universe wholly unlike our own, a place called Thule. It is a universe of pure shapes, formed as a perfect sphere and populated by polyhedra—a place idealized by Greek philosophers. But alas, this world was far from the idyllic paradise the ancient thinkers imagined, for the polyhedral denizens discovered that their universe was contracting, not expanding. This knowledge brought about much panic and confusion. Space itself became the most valuable commodity, and even in a world as abstract as Thule there resides such things as greed, envy, jealousy, and prejudice. More advanced polyhedra—those with more sides—began eliminating those less evolved than themselves. In the midst of this chaos rose a tetrahedron with high aspirations, which sought to prove that there were more sides to him than initially thought, and who would attempt to save their fragile world...



Bats=Death!

Example of Play

When a normal game begins, the player starts in the center of a large sphere. Immediately the sphere begins to slowly shrink. Groups of violet tetrahedrons spawn from multiple points on the inside surface of the sphere. They wander in flocks in various directions, and the player immediately begins flying back and forth, aiming and constantly firing his single-shot Geo-Cannon (gun). As the Geo-Shots hit the tetrahedrons, they explode in massive particle effects. They also spawn energy pulses that hover in space. As the player swoops in and out, up and down, clearing the enemies and picking up power-ups, the sphere expands noticeably, allowing more space in which to move.

Soon blue cubes and green octahedrons spawn along with the tetrahedrons. The player swerves around groups of tetrahedrons while dodging cubes and faster octahedrons that charge straight at the player. When one of the cubes is destroyed it drops a power pulse. The player immediately picks this up to upgrade his Geo-Cannon, which can now fire two Geo-Shots. Play continues in this manner for a few of waves of enemies, with enemies charging wildly and particles flying everywhere. All the while the sphere shrinks and expands, but it has generally been growing larger. The Geo-Cannon becomes able to fire multiple Geo-Shots at a time. The player spreads the fire out to efficiently disperse large groups of tetrahedrons, and focuses the gun narrowly to quickly eliminate the octahedrons.

After several waves tougher and more intelligent enemies such as the yellow dodecahedrons and orange icosahedrons enter the fray. The player must now dodge projectiles fired from both of these enemies, as the dodecahedrons spread out to cover a large area and a couple of icosahedrons chase the player from behind. Despite the numerous power-ups the player has acquired, this starts to prove more and more difficult, and every time he gets hit the spherical playfield jolts inward. The player has racked up a hefty score, but now he begins to feel the desperation as the sphere closes in around him. Every so often a glimmer of hope presents itself, but in the end the onslaught proves more than he can handle. The sphere collapses in on him with a monstrous burst of light and sound, ending the game.

Market Analysis

Target Market

The target audience for *Claustrosphere* is anyone ages 6 and up, and especially casual gamers. The gameplay will be simple, intuitive, and immediately rewarding to anyone.

Expected ESRB Rating

The development team expects a rating of E (Everyone) for *Claustrosphere*. The game will contain virtually no material content that anyone would consider offensive or that parents might find unsuitable for children. The only potential issue is the fact that the player is shooting and enemies explode when they are destroyed. However, the character designs and styles will be extremely abstract—to the point that they will make *Pac-Man* seem violent in comparison.

Competitive Analysis – Geometry Wars: Retro Evolved



<http://www.bizarrecreations.com/img/screenshots/gwre/index.jpg>

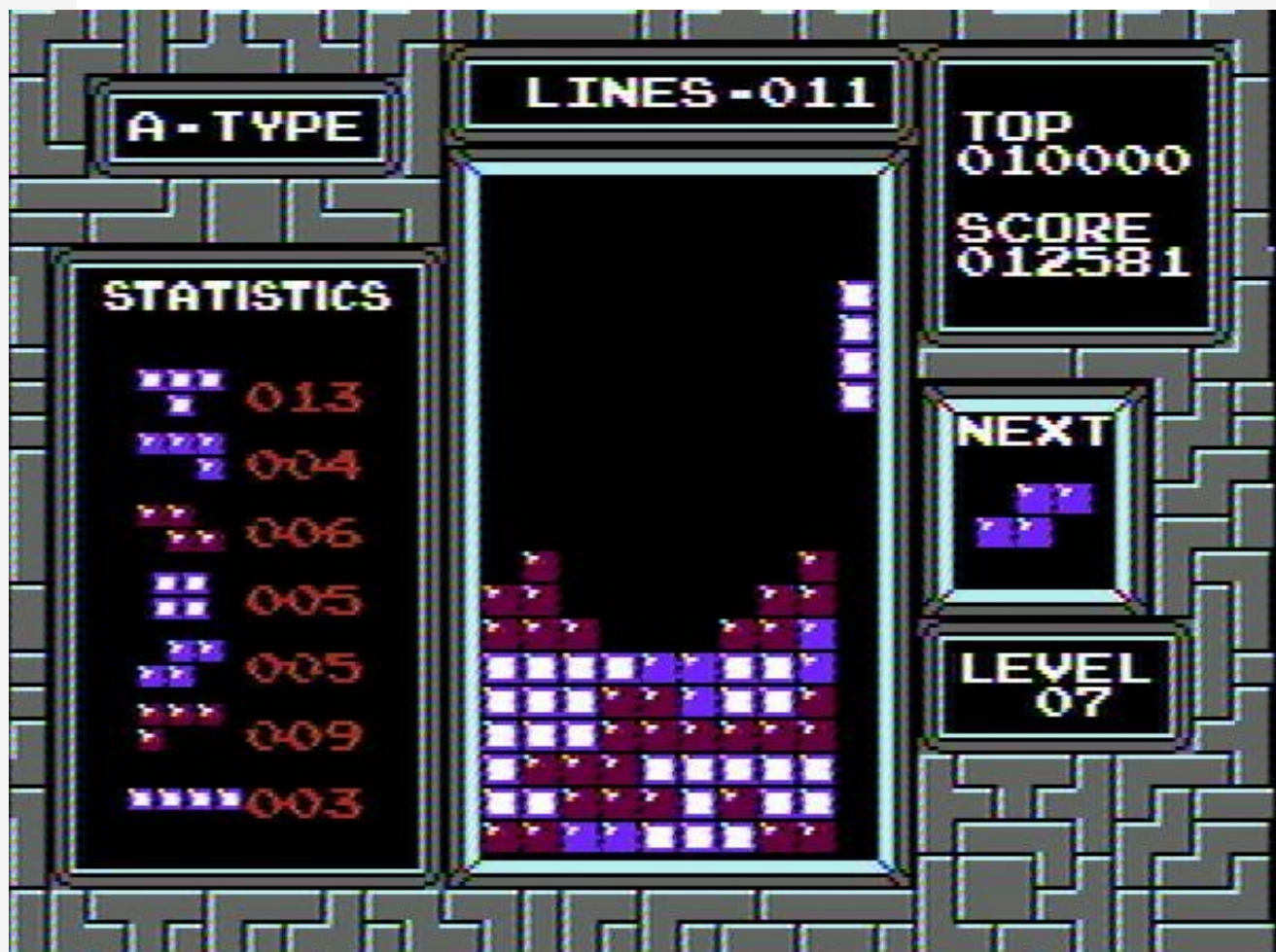
Geometry Wars: Retro Evolved (X-Box Live Arcade) is a heavy competitor due to its simple gameplay and flashy graphics. It is popular and easily distributed via X-Box Live Arcade. However, whereas *GW: RE* is a 2-D game, *Claustrosphere* adds an entirely new dimension that will take both the gameplay and graphics to the next level. The player will be able to fly around in full 3-D, the playfield will dynamically grow and shrink, and the visuals will feature spectacular lighting and texturing.

Competitive Analysis – Mars Matrix: Hyper Solid Shooting

http://media.dreamcast.ign.com/media/015/015621/img_1385638.html

Mars Matrix: Hyper Solid Shooting (Dreamcast, arcades) is another 2-D shooter game and potential competitor. It had unique gameplay features such as the ability to absorb enemy bullets and fire them back. One of the complaints about the game was its incredible difficulty level that made the game inaccessible to anyone but hardcore gamers. *Claustrosphere* will feature much more balanced difficulty levels that casual gamers will be able to enjoy. Furthermore, *Claustrosphere*'s graphics will be far more dazzling than the 2-D sprites featured in *MM: HSS*.

Competitive Analysis – Tetris



http://en.wikipedia.org/wiki/Image:Tetris_NES_play.png

Tetris (many platforms) is one of the most popular games of all time. It's simple and addictive gameplay appeals to a wide variety of gamers. We consider it a competitor because its target market in the casual games arena overlaps with the target market for *Claustrosphere*. The progression of difficulty and building of tension in *Claustrosphere* will be similar to *Tetris* as well. But whereas *Tetris* is primarily a puzzle-based game, *Claustrosphere* will incorporate large amounts of reflexive action, and, of course, the 3-D *Claustrosphere* will be much more visually appealing.

Bats=Death!

Cost Analysis

The projected development time for Claustrosphere is 9 months, and the expected budget is as follows:

Description	Cost	Number	Total
Programmer Salary	\$50,000	3	\$150,000
Programmer Workstation	\$2,000	3	\$6,000
Microsoft Windows XP Professional	\$300	3	\$900
Microsoft Visual Studio 2005 Professional Edition	\$1,200	3	\$3,600
Microsoft Office Professional Edition	\$500	3	\$1,500
			\$162,000

We expect a price point of \$20 for Claustrosphere. In order to make a profit, the game would have to sell approximately 8,000 copies. We expect the game to sell at least 25,000 copies, for a minimum profit of \$340,000.

Project Overview

Target Hardware Requirements

- **Operating System:** Windows 2000/XP
- **Processor:** 1.2 GHz Intel Pentium IV
- **Memory:** 64 MB RAM
- **Hard Disk Space:** 256 MB free
- **CD-ROM Drive:** 4X Speed
- **Video:** 128 MB VRAM with DirectX9.0c support, vertex and pixel shader versions 2.0 or higher
- **Sound:** DirectX 9.0c-compatible sound card
- **Input:** Mouse and keyboard or USB 2.0 controller
- **DirectX:** DirectX version 9.0c or higher

Milestone Schedule

The expected development time for Claustrosphere is 9 months.

First Semester Milestones:

- 9/11/06 – Game Pitch
- 9/29/06 – Game Design Document
- 10/13/06 – Technical Design Document/Project Timeline
- 10/23/06 – Revised Game Design Document
- 11/6/06 – Revised Technical Design Document
- 11/13/06 – Engine Proof
- 12/4/06 – First Playable

Second Semester Milestones:

- 1/19/07 – Revised GDD and TDD
- 2/9/07 – Pre-alpha
- 3/2/07 – Alpha
- 3/30/07 – Beta
- 4/13/07 – Gold Master

Team Members

- Joseph Boyd
 - Title: Producer
 - Primary Coding Responsibilities: Graphics, Physics
- Caleb Lee
 - Title: Technical Director
 - Primary Coding Responsibilities: Systems, Sound, Input
- Brandon Riggs

- Title: Designer
- Primary Coding Responsibilities: AI, Behaviors

Bats=Death!

Game Mechanics

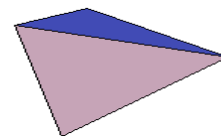
Game Walkthrough

After installing the game and starting it up, the player is greeted with the requisite splash screens, and then the title screen where the player need only press 'Enter' to start this experience. Once started, the player is introduced to the playing field and the game begins. At this point, waves of enemies are thrown at the player in rapid succession and the game continues until the sphere collapses on the player at which point, a "Game Over" screen is shown. At this point, the player is whisked back to the title screen where they can either quit the game or play again.

The Player

Character Design

The player's avatar is a very simple ship with lines accentuating the polygons that make up the player's model. It will have two wings and be a very light gray, almost white.



Movement

The player's movement is fairly quick and agile, but not so swift that players are unable to control the main avatar with any accuracy. The player's avatar is not tied to any particular plane, but will be free to move in any direction with the listed controls (for a full control list, see “[Controls – In Game](#)” in [Appendix B](#)). For the most part, the player should be faster or the same speed as the enemies.

Movement Action	Default Button
Change Direction	Mouse
Strafe Up	Space
Strafe Down	Left Control
Strafe Left	A
Strafe Right	D
Accelerate Forward	W
Rotate Counter Clockwise	Q
Rotate Clockwise	E

Abilities – Fire

From the beginning, the player will be able to fire their weapon, this will be fully automatic, and the rate of fire will be determined by the weapon's level described next.

Progression – Leveling

As the player progresses through the game, enemies he/she kills will have a small chance to drop a Power Pulse. When the player collects one of these elusive power ups, their weapon will level up permanently (to a certain maximum) in order to allow the player to cope with the ever-increasing quantity of enemies being thrown at them. For firing purposes, the player may either hold the fire button down, or they may choose to increase their Rate Of Fire (ROF) by manually triggering the fire button faster than their current ROF. The progression is as follows (note that all Shots Per Second (SPS) rates are subjective and will need to be tweaked to maintain playability and entertainment value):

- 0 - This is the starting level. At this point, the player shoots one bullet straight ahead at the rate of about 1SPS
- 1 - Increased rate of fire to 2SPS
- 2 - Increased rate of fire to 3SPS
- 3 - Increased rate of fire to 4SPS

- 4 - The player now shoots two projectiles at a time, both straight ahead, side by side, each individual stream now shoots at 2SPS and the player's overall SPS remains unaffected. However, through manual firing, their effectiveness has been doubled.
- 5 - Increased rate of fire to 3SPS per stream
- 6 - Increased rate of fire to 4SPS per stream
- 7 - Increased rate of fire to 5SPS per stream
- 8 - Increased rate of fire to 6SPS per stream
- 9 - The player now shoots three projectiles at a time, all straight ahead, side by side in a triangular pattern. Each individual stream now shoots at 3SPS and the player's overall SPS remains unaffected. However, through manual firing, their effectiveness is triple overall.
- 10 - Increased rate of fire to 4SPS per stream
- 11 - Increased rate of fire to 5SPS per stream
- 12 - Increased rate of fire to 6SPS per stream
- 13 - Increased rate of fire to 7SPS per stream
- 14 - The player can now shoot a spread shot with Maximum Deviation From Direction (MDFD) of 20° and a total SPS of 20.
- 15 - Each step from here on is just an improvement on this spread shot so I'm just going to list the stats as such: 20° MDFD, 22SPS.
- 16 - 20° MDFD, 24SPS
- 17 - 25° MDFD, 25SPS
- 18 - 25° MDFD, 27SPS
- 19 - 25° MDFD, 29SPS
- 20 - 30° MDFD, 30SPS
- 21 - 30° MDFD, 32SPS
- 22 - 30° MDFD, 34SPS
- 23 - 35° MDFD, 35SPS
- 24 - 35° MDFD, 37SPS
- 25 - 35° MDFD, 39SPS
- 26 - 40° MDFD, 40SPS
- 27 - 40° MDFD, 42SPS
- 28 - 40° MDFD, 44SPS
- 29 - 45° MDFD, 45SPS
- 30 - 45° MDFD, 50SPS
- 31 - 45° MDFD, 55SPS
- 32 - 45° MDFD, 60SPS

Bats=Death!

The Sphere (The Playfield)

The Sphere is central to the entire flow and dynamic of the game, and it serves multiple functions.

As Dynamic Playfield

The playfield of the game is the Sphere, and all action in the game is confined to the volume enclosed by the Sphere. The player has complete freedom to fly around anywhere inside the Sphere, and enemies will move around inside the Sphere. In the center of this sphere is an energy orb dubbed “the Centroid” that is insubstantial and does not stop the player's movement/momentum (see “[The Centroid](#)”).

The Sphere's radius will be constantly changing throughout the course of playing the game. The Sphere will grow and shrink depending on what happens in the playfield, what the player does, and how the enemies interact with the player. Initially, the sphere will start out at about ten times the player's diameter, this should give the player enough room to maneuver and come to grips with the controls, but be small enough to encourage the player to hurry up and start shooting enemies and make the sphere expand to give more breathing room.

The various events that will directly alter the Sphere's radius are:

- The sphere starts at an initial radius and with an initial decrement value that is subtracted from the sphere's radius per unit of time. This decrement value is also continuously decrementing per unit of time, the net result is a shrinking sphere that shrinks faster the longer it's around.
- When the player collects a Basic Pulse, the Sphere radius grows by a small amount.
- When the player collides with an enemy or an enemy projectile (anything that is considered to cause damage), the Sphere shrinks by an amount dependent on the strength of the enemy or projectile. E.G. - A player gets hit by a Polyhedron's projectile might cause the sphere to shrink by just the player's diameter, but if that same polyhedron kamikazes into the player, the sphere might shrink by three times the player's diameter.
- When the player is in contact with “the Centroid”, the sphere will shrink at a rate much faster than the normal shrinking rate.

As Health/"Game Over" Trigger

The player does not have any health or life bar in the game. Instead, the game is over when the radius of the Sphere is small enough that the player is forced into the Centroid and is in contact with both the Centroid and the Sphere. Thus the radius of the Sphere is a representation of the player's health.

Note that even though the sphere may be very small, the number of enemies spawned will NOT be changed, the point of this is to kill the player when they let the sphere get too small. As such, due to the mechanics of the game there will be a point where there is no conceivable chance of recovery as the sphere will be saturated with enemies. The player will be hitting enemies constantly thus making the sphere shrink that much faster and there will essentially be a chain reaction as the sphere will be collapsing on itself, forcing more enemies into the player to the point where the sphere is the same size as the player. At that point, an animation will be played of the sphere collapsing to nothing and the "Game Over" screen will be displayed.

As Score Modifier

The size of the Sphere directly influences the amount of points that the player scores when an enemy is defeated. The modifier is a C/R function, where R is the radius of the Sphere and C is a constant of proportionality. Thus, the smaller the Sphere, the more difficult it is for the player to maneuver, but the more points the player will score for defeating enemies. For more details, refer to [Scoring](#) section.

The Centroid

The Centroid is an insubstantial ball of energy that floats constantly in the center of the Sphere. It does not move or change size, but it serves two functions.

As Playfield Size Controller

When the player comes into contact with the Centroid, the Sphere starts shrinking at a faster rate than it normally does. This is to give the player control over their score multiplier if they feel the need to play in a masochistic way that will garner them a very high score.

As “Game Over” Detector

When the player is colliding with both the Centroid and the Sphere, the game should be over as there will be no way to collect enough Basic Pulses to counteract the expedited shrinkage of the Sphere that occurs from collision with the Centroid.

The Enemies

Claustrosphere's enemies come in a variety of polyhedral flavors. In addition to being visually distinct, the character models for different enemies will actually affect gameplay. For instance, an enemy's strength is directly proportional to the number of faces it has. More faces = More power. Bats = Death! The stronger an enemy is, the faster it is, the more damage it can receive, and the more damage* it can deal.

As far as specific stats for each enemy, see Appendix E.

*A Note on Damage. The player will not have hit points, per se, but the more damage an enemy does, the faster the rate of compression of The Sphere. For specifics, refer to "[The Sphere](#)" section.

Color Coordination

One of the goals in *Claustrosphere* is to relay as much gameplay information to player as possible without the use of a HUD. In adherence to this precept, then, the use of colors will play an important visual role. Enemies, in particular, will apply this color coordination. Each enemy will begin its existence at a particular color and will slowly fade to white as it receives damage. This will act as a very obvious visual indicator as to how much life remains.

Visual Effects

For the most part, the enemies will be pretty static, rigid models. Their coloring (as noted above) will be shifting as a visual indicator of their health, but other than that, while enemies are alive, they'll be pretty much just zipping around the screen. Some will be tumbling, but most will simply have a direction that's considered their nose and they follow it.

Tetrahedron (4 Faces)



Color: Purple

Overview: The tetrahedrons are the weakest and most basic of enemies. They are not very intelligent, but they do travel in packs. This often results in a simple flocking behavior (think: Follow the Leader). Tetrahedron will not go out of their way to attack the player. However, damage will be dealt to the player upon collision with them.

Bottom Line: Tetrahedron will act as a mere gun-fodder more than a serious threat. But they can be dangerous if the player does happen to get caught up in the middle of a swarm.

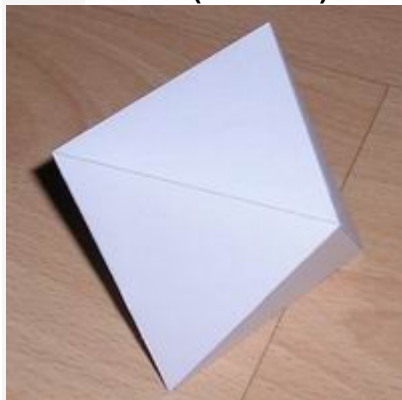
Bats=Death!

Cube (6 Faces)

Color: Blue

Overview: Cubes are not much more powerful than their 4-faced counterparts. They will employ a basic movement/patrolling pattern, albeit at a bit swifter pace than their Tetrahedral cousins and be a bit more erratic, thus being more dangerous than simple Tetrahedros.

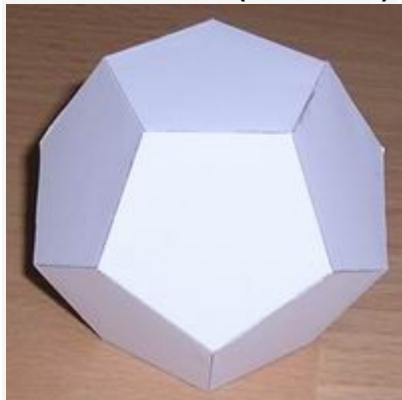
Bottom Line: These guys should not be taken to be a serious threat, but watch out when they group in numbers.

Octahedron (8 Faces)

Color: Green

Overview: Octahedrons are simple kamikaze artists that simply seek and destroy the moment they're born.

Bottom Line: When these guys are present, the player can expect a constant onslaught. Octahedrons are relentless.

Dodecahedron (12 Faces)

Color: Yellow

Overview: Dodecahedron is the first enemy that has the ability to fire projectiles at the player. They employ precise pathfinding techniques in an attempt to get at a desirable firing position. Additionally, they will often attempt to dodge incoming fire from the player. In this regard, Dodecahedron is the first enemy that can truly be considered "intelligent."

Bottom Line: These guys are smarter than any that the player has previously encountered. It will take a bit more strategy to bring them down.

Bats=Death!

Icosahedron (20 Faces)



Color: Orange

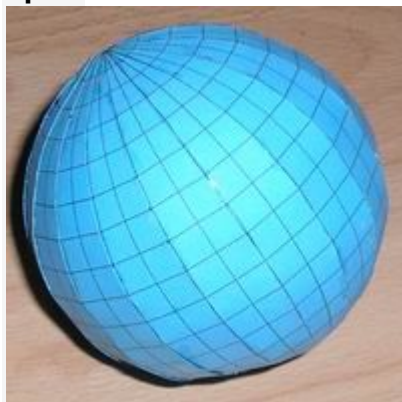
Overview: Icosahedrons are not very intelligent, but they are quite dangerous and a major nuisance to the player. Icosahedrons have a quick spreadshot with a decently fast reload time, but they become a MAJOR pain when the player realizes that they eat pulses.

Bottom Line: These guys are powerful, and tough. But don't be deterred. They must be stopped as quickly as possible so that the player may maintain a stable World Sphere.

The Ultimate Enemy

This brings us to the most formidable enemy found in *Claustrosphere*...the sphere.

Sphere



Color: Red

Overview: Up to this point, all enemies have been polyhedron with a finite number of faces. The sphere can be taken to be a regular polyhedral character encompassing infinite faces. In this regard, you can imagine that the sphere is very powerful. This enemy is undeniably fast, and immensely intelligent. It will employ behaviors found in many of the other "lesser" enemies, but has, in addition, its own unique power. Spheres can disappear at will, reappearing in a place they deem the most appropriate.

Bottom Line: Spheres are fast, intelligent, and very dangerous. It will take patience and strategic precision to defeat one of them.

Pictures in this section (above only) found from:

http://www.korthalsaltes.com/pictures_selection.html

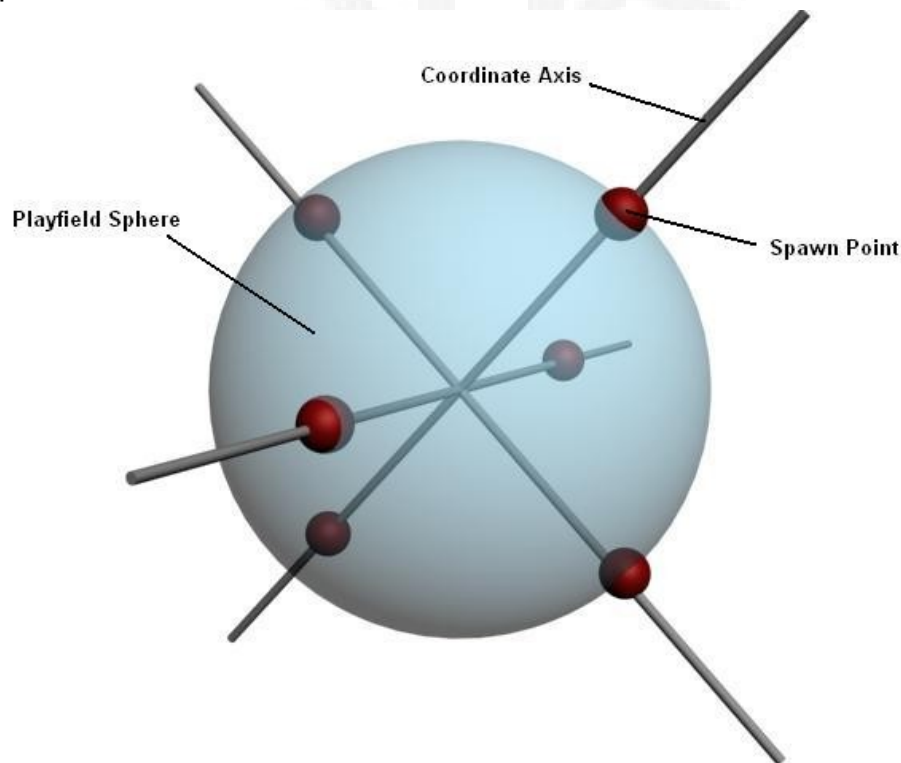
Enemy Spawning

Waves

The spawning of enemies will be algorithmically scripted. Enemies will come in “waves,” which are characterized by time. All enemies in a wave appear at approximately the same time. Once an entire wave has spawned, the next wave will not appear until a specific amount of time has expired. Initially, the time between waves will be relatively large, but as the game progresses, this time will decrease (in a $1/x$ style function). Every wave will have a variable, N , equal to the total number of enemies that will be spawned in the wave. N will increase over time, so that the longer the player survives, the more enemies will spawn per wave.

Spawn Groups

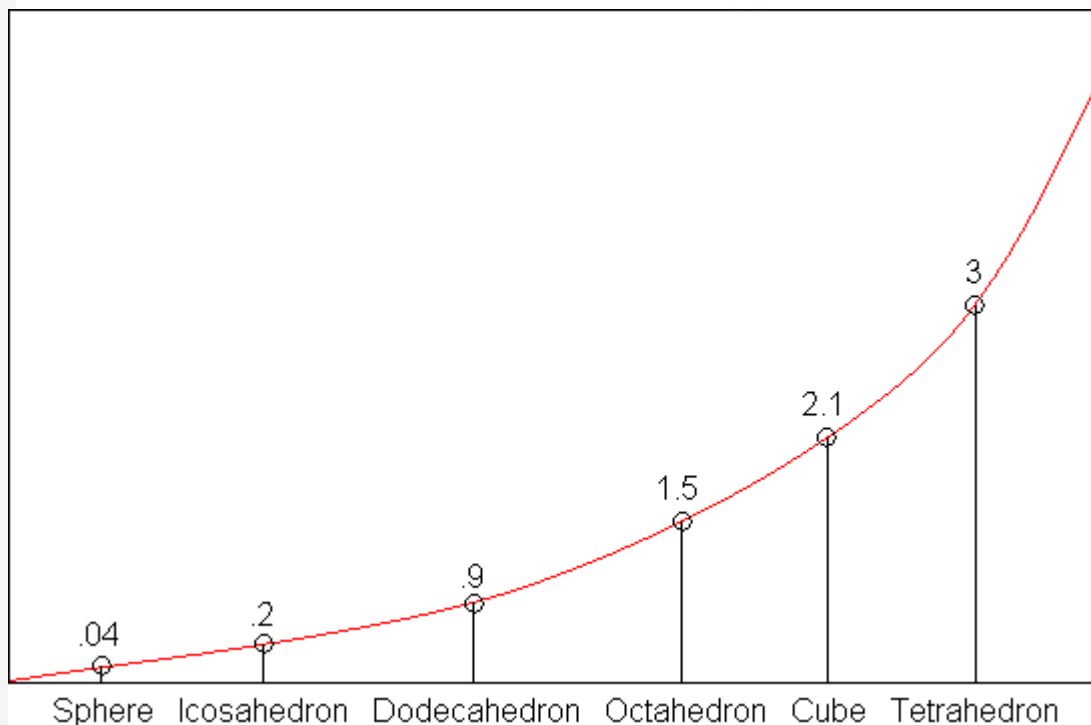
Each wave is composed of six “spawn groups,” which are characterized by location. The total number of enemies per wave will be evenly distributed among the six spawn groups. All enemies in the same spawn group originate from the same spawn point. The locations of these points are dynamically chosen for each wave by applying a random rotation to the world coordinate axes, whose origin is at the center of the sphere, and then choosing the six points where the rotated axes intersect the sphere. The following diagram illustrates the locations of the six spawn points:



Enemy Type Selection Process

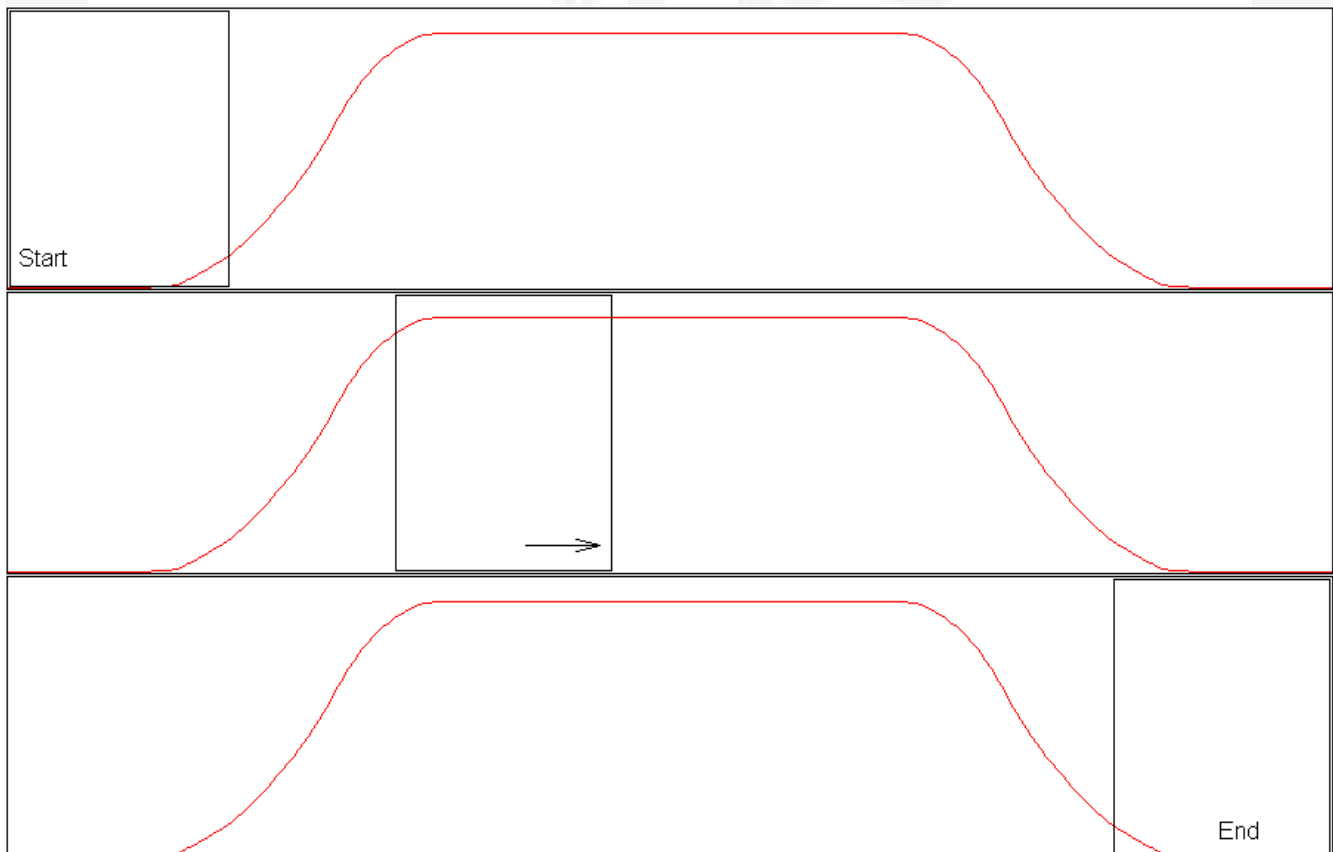
The enemies that are spawned in a particular wave will be selected based on probabilities set for each particular enemy type. The probabilities for the enemy types will be determined by using a bell curve. The process can be understood best by visualizing a moving window situated on a static curve. Different portions of the window will select the enemy types, with weaker enemies such as tetrahedrons found on the right and stronger enemies such as spheres found on the left (see diagrams below). The window will move slowly with time, exposing a different portion of the curve at any given time. Initially, the window will be situated so that only tetrahedrons are spawned. The curve will flatten out towards the middle of the game, resulting in a fairly even distribution of all enemies. Of course, further on in the game, the curve will produce probabilities highest for spheres.

Adding all the values on the curve for each enemy type, and then dividing each particular enemy's value by the group total will determine the probability for that enemy. A hypothetical situation, based on the diagram above, would produce the following probabilities:



Enemy	Weight	Probability	Lookup Values
Sphere	.04	.52%	1
Icosahedron	.2	2.6%	2-4
Dodecahedron	.9	11.6%	5-16
Octahedron	1.5	19.4%	17-35
Cube	2.1	27.1%	36-62
Tetrahedron	3.0	38.8%	63-100
Total	7.74	100%	

The calculated probabilities will determine a lookup table (the far right column above). When an enemy is created, a random number from 1 to 100 is generated, and the enemy type is then selected from the lookup table. In the above example, a random number of 45 would select a cube. This is repeated for each of N enemies in a wave.



Pulses (Pick-Up Objects)

All pick-up objects are called “Pulses”, and they come in two varieties. The player acquires Pulses by colliding with them. In order to make it easier to pick up Pulses, the player will siphon Pulses towards himself when they come within a certain “attraction range” from the player. If a Pulse is within this attraction range, it will accelerate toward the player.

Basic Pulse

Basic Pulses are small, white rings that emit particles of light produced when the player destroys an enemy. When the player picks up a Basic Pulse, the radius of the Sphere increases by a small amount. These Pulses are numerous, frequently appearing at the place an enemy is destroyed. The acceleration range for a Basic Pulse is moderate—a distance of approximately 7 times the size of the player. Basic Pulses will also have a moderate acceleration toward the player.

Power Pulse

Power Pulses appear on rare occasions when an enemy is destroyed. The player's guns are upgraded by one level when a Power Pulse is picked up. There is a very small chance that a given enemy will drop a Power Pulse, with the chances increased slightly for more difficult enemies. In general, the higher the number of polygons on an enemy, the higher the chance the enemy has of dropping a Power Pulse. The player will be required to maneuver with much more accuracy in order to pick up Power Pulses, and consequently their attracting range is small—a distance of approximately 2 times the size of the player—and their acceleration is slow.

Pulse Spawning

In general, Basic Pulses will be fairly plentiful and have a high chance of dropping from just about everything. However, Power Pulses can come from anything, yet they have a rather low chance of dropping (something like 1% chance, of course, this will need to be tweaked) from regular enemies, and higher chances from destroyed commanders. Basically, the tougher an enemy is, the higher a chance there is of spawning a Power Pulse.

Scoring

In most of the game modes, obtaining a high score is the objective of the game. As such, it is integral to the overall mechanics and is tied into a few different factors.

Base Score

The base score is determined by some point value assigned to each enemy and pickup in the game and the points are earned based upon killing the enemy and picking up power ups.

The scores are as follows:

Action	Base Score
Tetrahedron Killed	800
Cube Killed	1,200
Octahedron Killed	1,600
Dodecahedron Killed	2,400
Icosahedron Killed	4,000
Icosadodecahedron Killed	6,400
Sphere Killed	8,400
Pulse Picked Up	100
Power Pulse Picked Up	10,000

Sphere Modifier

The sphere modifies this score based on it's size, the initial value of the sphere should modify the score by nothing, whereas having a large sphere decreases the base score by a very small amount, but having a small sphere increases the score for each object by a VERY large amount. Something along the lines of a C/R function where C is some constant of proportionality and R is the current radius of the sphere. There will however be a lower limit to this modifier of .75.

Time Modifier

The time the player has been playing affects the score ever so slightly too, initially this value will be 1.0, but each time the spawn rates are increased/changed (See section "[Enemy Type Selection Process](#)"), this modifier will be increased by a very small amount (.01 for instance), thus there will be a ceiling for this modifier that will be determined by the number of steps in the Enemy Type Selection function.

Density Modifier

Take the number of enemies active, and divide that by the volume of the sphere ($\frac{4}{3} * \pi * r^3$). Now, multiply that number by some constant (to be tweaked at a later date) and you get your density modifier to the score. This modifier will only be applied to the score if it is greater than 1.0. This is to reward the player for fighting MANY enemies at once, but at the same time, not penalize them for keeping on top of the enemy spawning.

Time Without Damage Modifier

In order to reward the player for surviving in a playing field that is densely saturated with enemies, this modifier will kick in once the Sphere to Number of Enemies ratio reaches a certain threshold. This ratio is determined by using the same method as above ($\text{num enemies} / (\frac{4}{3} * \pi * r^3)$), but once this ratio hits a certain threshold (to be tweaked at a later date), then a new modifier kicks in, this new modifier is the “Time Without Damage” modifier. This modifier starts out at 2.0, but for every second the player survives without taking damage from enemies (note that collision with the Centroid does NOT affect this modifier as the player should be allowed to control the size of the sphere), this modifier is increased by 0.1. The only way to lose this modifier is to get hit by an enemy. Note that the threshold should be set sufficiently high so that this modifier is rarely (if ever) obtained as it will increase quite quickly and has no ceiling. Also, if the player manages to get things back under control after having this insanely high density, this modifier will continue to increase, this is by design to reward the player for achieving this ridiculous accomplishment.

Putting It All Together

Theoretically, the way this system should work is rewarding the player for both longevity and playing under self-imposed tension. To accomplish this, an enemy's birth score modifier will be calculated and involves the Time, Sphere, and Density modifiers. This modifier will then be changed over the course of the object's lifetime if a higher Time and Sphere modifier is obtained by the player. Then, when an enemy is killed, the Time Without Damage Modifier comes into play. For example: An enemy is spawned with a Time Modifier of 1.3 a Sphere Modifier of 2.2, and a Density Modifier of 1.0 which produces a modifier of $1.3 * 2.2 * 1.0 = 2.86$ and this is then stored with the enemy. Over the course of the enemy's life, the sphere shrinks to a Sphere Modifier of 4.8 at a Time Modifier of 1.5 and a Density Modifier of 1.5 (note that these values are NOT stored separately, thus the highest, simultaneous combination is needed, not the highest Time with the highest Sphere modifiers combined, the two values MUST occur simultaneously) giving the enemy a stored modifier of $4.8 * 1.5 * 1.5 = 10.8$. Then, the enemy is killed at a time when the Sphere Modifier is .8, the time modifier is 1.6, and the Density Modifier is 1.2 giving a total modifier of $0.8 * 1.6 * 1.2 = 1.536$. However, since 10.8 was the highest modifier obtained over the course of the enemy's lifetime, that is the modifier the player receives when they kill the enemy regardless of the current Time, Sphere, and Density modifiers. (Note that this highest modifier is stored not only for enemies, but also for Pulses and Power Pulses.) Then finally, the Time Without Damage Modifier is brought into the equation. For example, with the 10.8 modifier discussed above and a Time Without Damage modifier of 10.2, we would end up with a final score modifier of $10.8 * 10.2 = 110.16$.

Displaying the Score

The score will be displayed in the upper left corner of the screen and will usually be the only HUD element in-game. It will be very small and unobtrusive. However, when the game is paused, the score will be displayed in much larger fonts than those used in-game. It will be pretty prominent.

Special Effects

This belongs in the Mechanics section of our document because it is very much a part of the game and a driver for the feel of the game. This game will be absolutely laden with special effects. The reason for this is to just make so much happen on screen (or seem to be happening...) that the player gets a sense of total chaos that they somehow have control over!

The Sphere

The sphere should be a geodesic sphere inside of a sky-box textured with some deep space wall papers. The triangles of the sphere will be outlined in white, giving a wire frame impression and the transparent portions of the triangles should be tinged with a particular color relating to the player's current state (Red or Blue). When the player changes state, a cool graphical effect will take place which I will try to explain now...

The Centroid

The Centroid will be a glowing ball of energy that doesn't move or shift in the center of the screen. As the wave timer counts down, it intensifies in brightness/number of particles. Once the enemies are spawning, it immediately shoots out lightning in the six directions of the spawn points for the enemies and loses most of it's intensity and starts building up again. This cycle repeats indefinitely.

Sphere Size Change

The special effects here are nothing more than trying to accomplish a fluid change in the sphere's size with some exaggeration. When the sphere is shrinking, it will shrink beyond the new value a little bit and then wobble back into position. Same thing for when growing, the sphere will expand beyond the new radius, then "wobble" back.

Enemies

Particle effects, LOTS of particle effects. EVERYWHERE! When an enemy spawns, they will do so in a shower of particles. Catching a pattern? This mixed with the sheer number of enemies on screen will combine to create some crazy eye candy and chaos. When an enemy dies, they will light up anything around them (a light will be placed in the explosion that should dynamically light the surrounding environment).

More specifically, enemies are not textured and are just solid colors. However, they will have specular highlighting. When dying, enemies will spawn a shower of particles, and that's about it.

Projectiles

The projectiles in the game will be akin to glowing balls of energy that leave behind a trail of color based upon the spawning object. The ending explosion of the projectile will linger longer than the trail to give the impression of a final explosion. This explosion will be white when colliding with the world sphere, and if the projectile hits an enemy, this explosion will be the same color as the enemy's original color. Projectiles that are spawned from the player will leave behind a trail that is the same color as the player's state (Red or Blue), and projectiles

that are spawned from an enemy will leave behind trails that are the same color as the enemy's starting color.

The Player

Surprisingly, we don't have anything overly cool for the player.

Game Over

When the sphere reaches the same radius as the player's bounding sphere, the player will lose all control because the game is essentially over. At this point, the camera will zoom back to see the sphere quickly fall in on itself to the size of a pin prick, then explode in a super-nova, ejecting all the enemies held within into spectacular explosions of light.

Appendices

Appendix A – Interface Details

The following is a description of all the controls and what they do in-game, as well as the HUD.

Controls – In Game

The controls in this game will not be customizable.

Action	Button Assignment
Aim	Mouse
Fire	Left Mouse Button
Accelerate Forward	W
Accelerate Backward	D
Increase Shot Spread	Mouse Scroll Wheel Up
Decrease Shot Spread	Mouse Scroll Wheel Down
Strafe Up	Space
Strafe Down	Left Ctrl
Strafe Left	A
Strafe Right	D
Quick Flip (180° Turn Around)	Left Shift
Rotate Clockwise	E
Rotate Counterclockwise	Q

In Game – HUD

The ONLY things in the in-game HUD is the player's score and the current High Score. They will be very small and in the upper left corner of the screen. The score will be positioned in the upper left corner and starts at just "0" and goes up from there! The high score will appear just above that and be gray until the player reaches that high score at which point it will turn white and reflect the player's current score.

Appendix B – Project Schedule

Schedule Overview

- Engine Proof – 11/13/06
- First Playable – 12/4/06
- Pre-alpha – 2/9/07
- Alpha – 3/2/07
- Beta – 3/30/07
- Gold Master – 4/13/07

Engine Proof Checklist

- Modules
 - Input fully functional (controls working).
 - Graphics able to render 3-D models (models being shown on-screen).
 - Basic physics implemented (collisions being detected and reacted to).
 - Fundamental state machine functionality (objects are doing interesting things, like moving around).
 - Basic console functionality (debugging purposes).
- Classes
 - Window (the Window opening on your screen)
 - Game (the game object you will be playing)
 - Object (anything in the game, base class)
- Managers
 - Object – Loads object templates and creates object instances.
- Utilities
 - Memory Manager (internal usage to allocate space for everything)
 - Timer (used to time everything)
 - Vector (used to position/move everything)

First Playable Checklist

- Modules
 - Sound fully functional (you can hear music and sound effects).
 - Particle systems fully functional (you can see dazzling special effects when enemies are shot, shots are fired, etc.).
- Classes
 - Player (the object you are controlling)
 - Bullets (the things you are firing)
 - Sphere (the thing you are playing within)
 - All 3 Pulses (the objects you're picking up)
 - Tetrahedron (the four sided enemy)
 - Cube (the six sided enemy)
- Gameplay
 - Player able to fly around.
 - Player able to shoot bullets.
 - Enemies spawning.

- Score implemented.
- Sphere level dynamics implemented (sphere grows/shrinks over time and as a reaction to pulse pick-ups).

Pre-Alpha Checklist

- Player able to control cone spread (of your spread shot).
- Complete weapon upgrade system implemented (weapons can be upgraded).
- Enemies spawning in waves.
- Enemies implemented:
 - Octahedron (eight sided enemy)
 - Dodecahedron (twelve sided enemy)
 - Icosahedron (twenty sided enemy)

Alpha Checklist

- Splash Screens
 - DigiPen Logo
 - Team Logo
- Sphere enemy implemented (the little ball enemy thingy that's irritating as hell to kill).

Beta Checklist

- Most gameplay tweaking and balancing done.
- Most debugging done.

Gold Master Checklist

- Everything done!
- Green room visited!
- Alcohol consumed!

Bats=Death!

Appendix C – Enemy Stats

This section describes all the specific stats for each enemy. Now, keep in mind that these are all relative numbers and (of course...) will need to be tweaked in the final stages of production. Also, the damage done to the player is in terms of the player's diameter. So if a "1" is listed for Projectile Damage for an enemy, that means the sphere shrinks by an amount equal to 1 x Player's Diameter. But, the damage a player does (via their weapon mind you, if the player collides with an enemy, that enemy dies) is directly tied to an enemy's HP. That is to say that one bullet from the player removes one HP from an enemy. The Reload time listed is in seconds.

This is a quick description of the columns:

- HP: The object's hit points.
- Speed: The object's speed relative to the player.
- Kamikaze: The amount to shrink the sphere's radius by as a multiplier to the player's diameter when the enemy runs into the player. e.g. - ".8" equates to "shrink the sphere by .8 x Player's Diameter when this enemy runs into the player."
- Projectile: The amount to shrink the sphere's radius by as a multiplier to the player's diameter when the projectile hits the player. e.g. - ".8" equates to "shrink the sphere by .8 x Player's Diameter when this object's projectile hits the player."
- Reload: The time (in seconds) between the object's shots.

Description	HP	Speed	Kamikaze	Projectile	Reload
Player	N/A	1.0	∞	1 HP	
Tetrahedron	2	.75	.4	N/A	N/A
Cube	3	.8	.6	N/A	N/A
Octahedron	4	.9	.8	N/A	N/A
Dodecahedron	6	.4	8.0	4.0	1.5
Icosahedron	10	.9	2.0	1.0	.2
Sphere	32	1.2	6.0	3.0	.5

Appendix D - Team *Bats = Death!* Information



Joseph Boyd

Title: Producer

Primary Coding Responsibilities: Graphics, Physics

Originally from Atlanta, GA, Joe attended the Georgia Institute of Technology where he graduated Summa Cum Laude and received his Bachelor of Science in Physics in May 2001. Joe spent two years working for the Harvard-Smithsonian Center for Astrophysics in Cambridge, MA, and entered DigiPen Institute Of Technology in the Fall of 2003. He has major experience with physics engines—having written two in the past two years—and has done extensive coursework in physics and graphics. Joe's math and physics background should prove to be a valuable asset to the development of *Claustrosphere*, where he will be taking on the responsibilities of the graphics and physics engines.

Project History

- *Scat Attack* – Producer – Game Logic – 2-D puzzle game similar to *Pipe Dream* (NES).
- ***g*** – Technical Director – Systems, Physics – 2-D platformer in which the player can switch gravity and attack with a boomerang and bombs.
- *Medieval Kleptology* – Product Manager – Data Structures, Physics – 3-D platformer in which the player traverses a castle using a grappling hook and collects treasure.

Sign Off

Joe Boyd: _____



Caleb Lee

Title: Technical Director

Primary Coding Responsibilities: Systems, Sound, Input

Originally from Southern CA, and more recently Federal way, WA, he has been working in technical fields since early high school. Originally interested in networking and hardware, he dabbled in Electronics Engineering for a few years and then discovered a love of programming and as a result, now attends DigiPen Institute of Technology. This background in engineering has taught him the importance of attention to detail, no matter how mundane it may seem, and as a result has developed a real obsession when it comes to coding styles, and systems design. This quirk makes him a natural choice for the team's Technical Director as he tends to think things through in ways well outside and above the norm. He is currently working on improving his familiarity with the intricacies of C and C++ as he aspires to someday reach the status of "Code Guru" and is also focusing on Software Engineering to be able to better design systems and their interactions with other modules.

Project History

- *Syntax Attack* – Technical Director – Systems, Game Logic, Object Behaviors – Speed typing hybrid of *Missile Command* and a typing trainer.
- *Reveled* – Designer – Level Design, Data Structures – Innovative platformer with puzzle elements where the player builds the level around the character, akin to *Lemmings* (PC).
- *OGR* – Product Manager – Physics – Multiplayer FPS.

Sign Off

Caleb Lee: _____



Brandon Riggs

Title: Designer

Primary Coding Responsibilities: AI, Behaviors

For the development of *Claustrosphere*, Brandon will take on the roll as game designer for the group known as *Bats = Death!*. Mr. Riggs' influence in the gaming industry comes from his most recent role as the game designer and physics programmer on *Emetic Games' Once Knight*. In addition to this role, he is perhaps better known for his Artificial Intelligence and behavior algorithms implemented in his two prior games, *Fussy Chaps' g*, and *Disco Rockxxx's Stayin' Alive*. He achieves realism by creating enemies with different degrees of "Intelligence," and a wide array of behaviors. Brandon now lives in Redmond, WA with his beautiful wife and two kids, and is a part-time employee at *Nintendo Software Technology*. He grew up in Las Vegas, NV where he has played video games his entire life. With the experience Mr. Riggs has to offer, he will be a great asset to *Bats = Death!*.

Project History

- *Stayin' Alive* – Product Manager – Level Design, Enemy Behaviors – 2-D action shoot 'em up in the spirit of *Smash TV* (arcade).
- **g** – Producer – Behavior Systems, AI– 2-D platformer in which the player can switch gravity and attack with a boomerang and bombs.
- *Once Knight* – Designer – Physics – *Four-Swords* (Gamecube) inspired multiplayer action-adventure game.

Sign Off

Brandon Riggs: _____