# TABLE OF CONTENTS

1.0.0:: THE TEAM  

2.0.0:: GAME BUILD HISTORY  

3.0.0:: GAME OVERVIEW  

## 3.1.0:: "In a Nutshell"  

## 3.2.0: Common Questions  

3.2.1:: What is the game?  

3.2.2:: Why create this game?  

3.2.3:: When/where does this game take place?  

3.2.4:: What is the user in control of?  

3.2.5:: What is the user’s main goal?  

3.2.6:: How does the user win?  

## 3.3.0:: The User Experience  

3.3.1:: Genre  

3.3.2:: Core Game Play  

3.3.3:: Game Play Example  

## 3.4.0:: The Platform  

3.4.1:: The Platform  

3.4.2:: System Requirements  

3.4.3:: Game Engine  

## 3.5.0:: Target Audience  

## 3.6.0:: Game Time  

3.6.1:: Total Game Length  

3.6.2:: Replayability  

## 4.0.0:: BASIC CONCEPTS  

## 4.1.0:: Storyline  

4.1.1:: Background Story  

4.1.2:: In-game Story  

4.1.3:: Victory Conditions  

## 4.2.0:: Heroes  

4.2.1:: William Gentry  

## 4.3.0:: Enemies  

4.3.1:: Rockimichili  

4.3.2:: Aquacivi  

4.3.3:: Fistucko  

4.3.4:: Stocreep  

4.3.5:: Siffler Swarm  

## 4.4.0:: User Interface  

4.4.1:: Targeting Reticule  

4.4.2:: Health Meter  

4.4.3:: Chroma Cycle  

4.4.4:: Chaos Meter  

4.4.5:: Chaos Cycle
4.4.6:: Score

4.5.0:: Distinguishing Features
4.5.1:: Chroma Blasting
4.5.2:: Chroma Cycle
4.5.3:: Chaos Cycle
4.5.4:: Available Power-ups
4.5.5:: Point Distribution

4.6.0:: Controls
4.6.1:: Control Diagram

5.0.0:: NAVIGATION CHART
5.1.0:: Start Menu
5.2.0:: Escape Menu
5.3.0:: Options Menu
5.4.0:: Level Movement

6.0.0:: SCENES AND ACTION
6.1.0:: Rockimichili’s Temple
   6.1.1:: Fountain Room
   6.1.2:: Pit Room
   6.1.3:: Tower Room
   6.1.4:: Mirror Room
   6.1.5:: Pole Room
   6.1.6:: WooWoo Battle
6.2.0:: Temple Escape
6.3.0:: Amazon Jungle
6.4.0:: Cave
6.5.0:: Desert Valley
6.6.0:: Balbaringer Mountains

7.0.0:: RESOURCES
7.1.0:: Models
   7.1.1:: Gentry
   7.1.2:: Rockimichili
   7.1.3:: Aquacivi
   7.1.4:: Fistucko
   7.1.5:: Stocreep
   7.1.6:: Siffler Swarm
7.2.0:: Sound Effects
   7.2.1:: Aquacivi
   7.2.2:: Fistucko
   7.2.3:: Stocreep
   7.2.4:: Siffler Swarm
   7.2.5:: Gentry
   7.2.6:: Environment
   7.2.7:: Music
7.3.0:: Backgrounds
   7.3.1:: Main Menu
   7.3.2:: Loading
   7.3.3:: Death
7.5.0:: Milestones

SAKE OF CLARITY PRODUCTIONS
7.5.1:: Animation 60
7.5.2:: Modeling 60
7.5.3:: Presentation 60
7.5.4:: Programming 60-61
7.5.5:: Menus 61
7.5.6:: Audio 61
7.5.7:: Documentation 61

8.0.0:: APPENDIX 62-89

8.1.0:: Programming 62-71
  8.1.1:: Action 62
  8.1.2:: Function 62-63
  8.1.3:: Organization 63
  8.1.4:: Time 63
  8.1.5:: Player 63-64
  8.1.6:: Fire 64
  8.1.7:: Enemies 64
  8.1.8:: What knows where what is and why? 64-65
  8.1.9:: Particles 65
  8.1.10:: Score 65
  8.1.11:: Camera 65
  8.1.12:: Music 66
  8.1.13:: Sound Effects 66
  8.1.14:: Chroma Cycle 66-67
  8.1.15:: Health Bar 67
  8.1.16:: Chaos Cycle 67
  8.1.17:: Intractable Objects 67
  8.1.18:: Mokia Box 67-68
  8.1.19:: Animations 68
  8.1.20:: Creature Destruction (Wrath of God) 68
  8.1.21:: Gravity? 68
  8.1.21:: Enemy AI Turning 69-70
  8.1.23 :: Perfect Circular Angle Theory 70-71
  8.1.24:: Useful Information 71
  8.1.25:: Resources 71

8.2.0:: Importing Animation into GS 72
  8.2.1:: Max to Maya 72
  8.2.2:: From Maya 72-73
  8.2.3:: Triggering the Animation in GS 73

8.3.0:: External Object Texturing into GS 74

8.4.0:: Texturing w/GS 75-76

8.5.0:: Sound Effects 77
  8.5.1:: Sources 77
  8.5.2:: Creation 77

8.6.0:: HUD 78-79
  8.6.1:: Requirements 78
  8.6.2:: Techniques 78
  8.6.3:: Layering the HUD 78-79

8.7.0:: Journals 80-89
  8.7.1:: Matt Parent 80-82
  8.7.2:: Matt Smith 82-83
  8.7.3:: Ari Teger 83-84
  8.7.4:: Alex Tanzio 85-86
8.7.5:: Pat Xin 86-87
8.7.6:: Josh Landis 88-89
1.0.0 :: THE TEAM

Matt Parent :: Associate Producer | Writer | Audio  
Matt Smith :: Lead Animator | Programmer  
Ari Teger :: Lead Modeler  
Alex Tanzio :: Lead Programmer  
Pat Xin :: Lead Art Director | Modeler | Programmer  
Josh Landis :: Modeler | Animator

2.0.0 :: GAME BUILD HISTORY

1.0.0.0: 
Implemented general movement with keys and mouse  
Implemented general movement of 3rd person camera

1.0.0.2: 
Implemented AI script

1.0.0.3: 
Implemented the firing of a weapon  
Implemented the firing of a weapon with chroma wheel timing

1.0.0.4: 
Inserted all enemy models into the game

1.0.0.5: 
Inserted a general model for Gentry

1.0.0.6: 
Inserted the world map for our prototype level

1.0.0.7: 
Applied enemy models with animations (walk, attack)

1.0.0.8: 
Applied Gentry with animations (walk, attack, push, pull)

1.0.0.9: 
Added sound to everything (enemies, music, environment)

1.0.1.0: 
Implemented our HUD so that the Chroma Cycle was fully operational

1.0.1.1: 
Implemented the code for score to be calculated
1.0.1.2: Implemented combo kills when nearby enemies of the same color are killed within a certain radius

1.0.1.3: Implemented various puzzles to the game

1.0.1.4: Implemented triggers that destroy the enemies in the previous room upon exit of that room

1.0.1.5: Refined the camera movement

1.0.1.6: Added particle effects to Gentry’s attack as well as enemies dying

1.0.1.7: Redecorated the entire level with bones, torches, and the like

1.0.1.8: Added lights to the game

1.0.1.9: Added a new HUD with updated Chroma Cycle and Chaos Cycle
3.0.0 :: GAME OVERVIEW

3.1.0 :: “In a Nutshell”
Chroma Chaos is a 3rd person, fast, puzzle-based, adventure game with a colorful twist, which has your character traverse ancient temples, treacherous jungles, and arid deserts while combating multi-colored creatures with chaotic intent.

3.2.0 :: Common Questions
3.2.1 :: What is the game?
Chroma Chaos is a game in which your character, named William Gentry, has been imbued with a special power to shoot different colored blasts from his hands. You must blast the right colored enemies while simultaneously executing puzzles to get to the next room. In the end you must obtain the sacred WooWoo nut.

3.2.2 :: Why create this game?
Chroma Chaos must be created because of its combination of simplistic control scheme, engrossing story, and dynamic gameplay, which has the user challenged on two different levels for the game’s entirety.

3.2.3 :: When/where does this game take place?
Chroma Chaos takes place in a couple of different regions. Your character will have the chance to trudge through ancient temples, humid rainforests, moist caves, arid deserts, and suffocating mountain peaks. The game does not take place during contemporary times, but is cemented within the time period during the turn of the 20th century. Our main character is from the upper English middle-class and his demeanor as well as his clothing will reflect his stature.

3.2.4 :: What is the user in control of?
The user is in constant control of William Gentry, an English archeologist in search of the sacred WooWoo nut. The user is control of him through a 3rd person vantage point with a reticule in the center of the screen to aid in lining up successive chroma blasts. In addition to controlling his aiming the users are able to push/pull, dodge, jump, and solve puzzles all with one simple button, which is dependent on what situation you are in. With this simplicity Sake of Clarity wants to incorporate younger audiences into the fun.

3.2.5 :: What is the user’s main goal?
The user’s main goal is to obtain the WooWoo nut from the top of the Balbaringer Mountains. However, before they get there, the
user must battle through a number of different regions, each with a “room-to-room” puzzle mentality to them. For instance, during the first level the user must solve puzzles and traverse room-by-room in order to get to the next level. Puzzles are predominantly based off of colors you can shoot from your character. The puzzles will also increase in difficulty as your character moves through the game.

3.2.6 :: How does the user win?
The user will win once they lead Gentry up the Balbaringer Mountains, defeat Rockimichili (an ancient God of War), and find the WooWoo nut. However, the user will also have a sense of satisfaction with defeating each level’s main boss as well as even defeating each room’s puzzle. This sense of accomplishment will keep the user glued to the screen to see what happens next.
3.3.0 :: The User Experience

3.3.1 :: Genre
Chroma Chaos is a distinct blend between many popular gaming genres. The game’s perspective revolves around the 3rd person adventure game. The back of the character is seen at all times during game play, which broadens the view for the gamer. The user will be able to scan the room and figure out what has to be done next. This view also helps with movement as the user more easily can jump and dodge objects as they present themselves. The first person shooter is also used by using a targeting reticule that is moved around much like the reticle in the FPS genre. A third genre is adopted by adding the element of puzzle solving. The user will continually have their brain challenged by color-based puzzles under pressure by the game’s many monsters.

3.3.2 :: Core Game Play
The core of this game is a balance between battling back a barrage of enemies while simultaneously figuring out the puzzle in the room. The gamer will have to use the Chroma Cycle in order to match up different colored blasts to the corresponding color of the enemy they want to defeat. The user will sometimes also be given a chance to block the flow of the enemies by utilizing different environmental obstacles. Once the user gains access to the next area of the game they will be given a different puzzle to solve. The game proceeds in this fashion until the user fights his way to each level’s boss battle. After the user defeats the boss they will be able to gain access to a different region with the game’s universe, which ultimately leads to the Balbaringer Mountains— the final level.

3.3.3. :: Game Play Example
You come upon a four-sided room with stone architecture encompassing the whole of it. A quick scan of the room shows two enormous holes on each side of the room, three different colored rocks in various locations, and two different colored locks that flank a door that reaches the ceiling.

Before you can take another step you hear a low rumbling, which at first starts at a distance but exponentially grow closer with each passing second. The low rumbling is now close enough that it is mixed with high-pitched, animal-like screams. Within seconds multitudes of multi-colored enemies come pouring out of the holes in the wall. Quickly, you turn around and shoot off a quick blast in multi-colored succession, destroying the first wave. One of your yellow shots misses its target and hits one of the colored rocks in the room, which launches it towards the hole in the wall, where the enemies are emerging from. This massive rock destroys all enemies
within its path as it quickly seals up one of the two holes. Quickly realizing that you can move these blocks with your powers you send a wave of chroma towards another rock at the other end of the room. The block misses the hole in the wall and you are forced to dodge enemies that come inches away from ending your day. Another concentrated burst of chroma takes care of these creatures in close proximity and allows you to seal up the last hole in the wall.

With enemies howling and snarling from behind the rock, you make your way over to the two multi-colored locks keeping you from further exploring this treacherous temple. With two quick bursts of chroma to the respectively-colored locks the massive door inches upon, leading you to further exploration in an attempt to gain the WooWoo nut.
3.4.0 :: The Platform

3.4.1 :: Platform
Chroma Chaos will be available on the PC.

3.4.2 :: System Requirements
Pentium 700 MHz
256 MB RAM
32 MB graphics card
Windows 98/ME/200/XP
100 MB HDD

3.4.3 :: Gaming Engine
Chroma Chaos will be created using GameStudio.
3.5.0 :: Target Audience
The Sake of Clarity team wants this game to be easy enough to pick up by anyone that wants to play it. The game will have only the most minimalist of controls, utilizing just three primary buttons on the keyboard and mouse.

The age group that our production team is aiming for is between 7 and 12 years old. The way we plan to grab the attention of this age group is by the aforementioned ease of use, but more importantly the look and feel of the game. The way Sake of Clarity plans to achieve this is through the use of four distinct colors- blue, red, yellow, and green. By using these vibrant and popular colors the game will have a cartoon-like outlook as well as boasting easily distinguishable targets, which will make it easier for the amateur gamers of this age group. The characters in the game will also crack jokes that are targeted to this age group, which will keep them enthralled in the story to see what Gentry says next.

However, people of all demographics will be able to enjoy this game, because of its game play depth. Games like Legend of Zelda Wind Waker, Katamari, and Viewtiful Joe all have a cartoonish look to them but have become popular with all age groups even though it is targeted towards younger audiences. Wind Waker was the #5 best selling game of 2003 and sold more than 1.9 million copies in the United States alone! Game sites everywhere have been raving about the unique style of game play that Viewtiful Joe unleashed. Gamespot gave this title a 9.2, pointing at its different direction. This is something that we want to do with Chroma Chaos. Never before has a game been made that sets out to do what we’re doing.

For players who seek more difficult challenges of game play there will be adjustable difficulty levels to fine tune the game to your tastes. The game will also not contain any amount of gore or any element that would designate a rating above “E” for everyone.

Resources:
http://cube.ign.com/articles/613/613714p1.html
http://retailindustry.about.com/od/seg_toys/a/bl_npd012703_2.htm
3.6.0 :: Game Time

3.6.1 :: Total Game Length

Note: These game times do not take into consideration the amount of replay-ability from this game, which will take the user perhaps triple the amount of their experience level’s time.

Amateur Gamer
- Temple = 30 minutes
- Temple Escape = 15 minutes
- Amazon Jungle = 35 minutes
- Cave = 30 minutes
- Desert = 40 minutes
- Balbaringer = 45 minutes
- TOTAL = 195 minutes

Casual Gamer
- Temple = 25 minutes
- Temple Escape = 10 minutes
- Amazon Jungle = 25 minutes
- Cave = 25 minutes
- Desert = 30 minutes
- Balbaringer = 40 minutes
- TOTAL = 155 minutes

Experienced Gamer
- Temple = 20 minutes
- Temple Escape = 8 minutes
- Amazon Jungle = 20 minutes
- Cave = 20 minutes
- Desert = 20 minutes
- Balbaringer = 35 minutes
- TOTAL = 123 minutes

3.6.2 :: Re-playability
Chroma Chaos will be re-playable through numerous facets. The strongest asset for getting players to play the game once they have beaten it, is give them a chance to replay the game with a certain added feature. For Chroma Chaos there will be numerous unlockables, which can be obtained through good performance. These unlockables will then be available the next time the player starts up a game, or if good performance in a level is merited they will obtain the unlockable during the next level. Unlockables come in the form of power-ups, which will be discussed in their specific section. The player can also unlock different difficulty modes, which gives them more chances to acquire different power-ups.
However, this re-playability cannot rely on unlockables alone. It also needs to continue to challenge the user each time they play the game. This facet can be obtained with the addition of different difficulty levels. The user may tailor their game to their initial level, and then once they beat the game, they can up the difficulty to continuously challenge them. The game will be more difficult through a more abundant and faster mass of enemies, and less ways to impede their progress towards you. Each difficulty set will have different unlockables to strive for, which challenges the user to complete the game on all difficulty levels.
4.0.0 :: BASIC CONCEPTS

4.1.0 :: Storyline

4.1.1 :: Background Story

Buried deep within the South American jungle near present day Peru lived a Native American tribe named the Merdunk. Not much is known about this ancient people who lived approximately one thousand years before the great Mayan Empire. Whatever is known about this great civilization is from the Mayans who settled upon the Merdunk’s ruins. For instance, it is known that the Merdunkians were primarily a planting society; planting miles and miles of maize, beans, and other vegetables. It is also known that the Merdunkians experimented with a wide range of colors from nearby flora. They used the different colored flowers of the area to stain their clothing as well as weaponry. Before attacking their enemies they would ravenously gather up all of these special flowers, mash them all together in a pool of water, and bath themselves in the radiant irrigate. The outcome of this ritual is not fully known or understood, but it has been said that the Merdunkian warriors were aided in their struggles against neighboring tribes. Because of the overuse of these flowers they have long since become extinct.

Aside from this rare flora there were other rare species of plant life that intrigued the Merdunkians. Warriors would strive for it through centuries of the Merdunkian’s history. The nut of the WooWoo tree was the cause of many deaths and only one glorifying victory. The nut is native to the Balbaringer Mountain Range in what is now known as the Andes Mountains. Living in perhaps one of the harshest climates in the world, this nut would become the aim of many Merdunkian warriors. Each soldier was given one chance in their lifetime to climb the Balbringer Mountains and bring back this sacred nut. When a warrior brought back a WooWoo nut they were thrown a massive parade and a temple was erected in their name so that their spirit could be worshipped long into the afterlife. With the possession of this nut a person could successfully capture a certain god-like stature among the people of
the Merdunk Empire. Only one man was successful at obtaining the nut and his name was Michitalon. The temple erected in his name although documented by the Mayans has never been found however.

The Merdunk Empire also is filled with countless accounts of spiritual disturbances, which have been noted by Mayans through their hieroglyphic symbols. Stories of monstrous creatures attacking outlying villages as well as damaging whole fields of crops terrified the Mayan people for centuries. The creatures depicted on walls all over Mayan society don’t capture everyday serpents, gorillas, dogs, and other well known creatures to the area. What they do capture is somewhere between these species and another world. Armies of brightly colored creatures have swept through entire Mayan towns and have vanished just as quickly as they have come. The fall of the great Merdunk Empire perhaps could be attributed to these dangerous beings.

Hundreds of year’s later one man William Gentry will seek to find that nut.

This is where Chroma Chaos begins.

4.1.2 :: Prototype Level Script
This mission starts off with a cut scene. The animation will start with our reluctant hero; William Gentry cuts through the jungle foliage. Gentry then spots the temple off in the distance. He then runs toward it, entering the temple. The opening room is given a
fly-by with the camera, detailing the ancient temple’s architecture, and the mesmerizing pool located in the center of the expansive room. Gentry remarks, “I’m quite parched” and removes his hat, revealing a tea set hidden beneath it. He walks up to the pool to take a drink, but after tossing a cup back, his hands begin to convulse. He then is shocked, when multi-colored blasts begin to erupt from his palms. As this happens, bizarre creatures erupt from a gaping hole in the temple wall. Gentry realizes, as the colored blasts hit a number of the creatures, that the monstrosities hit with the same color as their aura, are destroyed. With the secret in hand, he then braces himself to defend from the onslaught.

In order to pass to different rooms you must utilize varying amounts of strategy. In each room there is a puzzle which the player must solve in order to pass to the next room. While doing so the player must continue destroying enemies as they flow from different places in the room. Once an opportunity presents itself, the user may have the option of blocking off the entrance to where the creatures are coming out of. In this instance once the pathway is blocked they cannot attack you and the player is allowed to solve the puzzle uninterrupted. The player also has the option to rack up points by destroying all the enemies in a room, which will give maximum points. There are a certain amount of enemies to each room, so they will run out.

After battling countless amounts of enemies and figuring out all of the first level’s puzzles you will enter a huge room in which the WooWoo nut sits in the very center. Just as the player gets close to this sacred artifact Rockimichili will appear rearing four different heads.

Rockimichili utilizes two different kinds of attacks from each face. One will be able to be deflected back at him, while the other attack will be avoidable only by hiding behind the different blocks that litter the room or hitting the action button to dodge. For example, Rockimichili might throw different colored rocks at you and if you successfully hit the rocks with the right chroma they will hurt Rockimichili. However, Rockimichili’s eyes might start glowing and a powerful laser blast will injure you if you don’t hide behind a block or dodge it. Once a certain number of hits are made Rockimichili will discontinue the fight, but not after stealing the WooWoo nut from the room and taking off down a hallway, which is the setting for the next level.

4.1.3 :: Victory Conditions
Victory conditions can be broken down into three different forms. The player will obtain small victories by figuring out each room’s
puzzles and moving on to the next. Once the player defeats each level’s boss they will have achieved a much larger victory by completing the level. All the levels are building to the ultimate victory when Gentry faces off against Rockimichili for one final confrontation on top of the Balbaringer Mountains. If Rockimichili is defeated Gentry gains access to the WooWoo nut and the game and the victory conditions have been satisfied.
This undistinguished, British archeologist is good natured and certainly does mean well with his actions, however he is rather unsuccessful. With his long, scruffy moustache, his classic safari hat, and his wise cracking British accent- it is hard not to laugh while he is on screen. Through various cut scenes and levels the player follows Gentry on his journey to gain recognition by returning home with the WooWoo nut, successfully ridding himself of his career embarrassments.

**Damage = 10% each hit**

**Boss Damage = 15% regular attacks**

25% attacks that you can send back
This is the ancient Merdunkian god of war, which will persistently get in the way of Gentry on his journey for the WooWoo nut. First taking the shape of a multi-faced rock formation this boss will take careful planning of shots to take down. Through advancing levels the player must be cautious as Rockimichili has the ability to take on many different characteristics of his surroundings to successfully take down Gentry and put to rest the discovery of the Merdunkian Empire for centuries to come.

**Damage = 12.5%**

**Speed = ½ Gentry**

**Rate of Fire = 4-5 seconds (rotates every 2-3 shots)**
4.3.2 :: Aquacivi

The Aquacivi are blue in color and are very fluid in motion and appeal. These serpents will cut through the air as if they are swimming unobstructed through the water. The only way to send these blue demons back to their demise is by hitting them with a blue blast from your chromanized hands.

**Damage** = 100%

**Speed** = 75% Gentry

**Attack Radius** = 70 cm
These red, fire-churning dogs will not let you get away with the WooWoo nut without a fight. These minions will steam across the temple floor in pursuit of their sacred necessity in your possession. The Merdunkians kept many domesticated dogs on their farms to scare away birds as well as warn of approaching enemies. In the afterlife they have been possessed with the chroma and exist for defending the WooWoo nut. Kill them with a blast of red.

**Damage** = 100%

**Speed** = 90% Gentry

**Attack Radius** = 80 cm
4.3.4 :: Stocreep

These descendents from Rockimichili are covered in a thick, grassy moss, which harbors the moisture of the chroma. These rock slabs move slowly but are exceptionally strong in battle. A quick blast with green chroma will put an end to their existence.

**Damage** = 100%

**Speed** = 50% Gentry

**Attack Radius** = 100 cm
4.3.5 :: Siffler Swarm

These small golden beetle-like creatures band together to form a suffocating swarm. Once these bugs are upon you there is little-to-no escape as they will nibble and suffocate. It is best to eliminate them with a strong blast of yellow chroma as fast as you can.

**Damage** = 100%
**Speed** = 65% Gentry
**Attack Radius** = 80 cm
4.4.0 :: User Interface

4.4.1 :: Targeting Reticule
The targeting reticule and the whole system of aiming are taken from other 3rd person shooters such as Max Payne and the MechWarrior series. In the middle of the screen there is a simple reticule made up of four perpendicular lines. As the player moves the mouse around the reticule will stay in the center of the screen as the camera moves in respect to the player’s mouse movement.

If the corresponding chroma of an enemy is highlighted in the Chroma Cycle and the player moves the reticule over an enemy with this chroma the reticule will change value to white. This will also occur if an object that can be moved is targeted and its corresponding chroma is highlighted in the Chroma Cycle. During times when there isn’t a correlation
targeted enemy or when nothing is targeted, the reticule will be black.

4.4.2 :: Health Meter

Chroma Chaos’ health system is unique in its representation. Instead of the typical numerical system to identify your health there will be a graphical depiction of a stone head. When the player has full health the head will have his eyes and mouth wide open, however as the player obtains damage the totem will slowly close his eyes and mouth. When the player is at 0% health the totem’s eyes and mouth will be completely closed and the player will have to restart the current room.

4.4.3 :: Chroma Cycle

The Chroma Cycle should appear as a stone-like wheel with multiple colors that make it up. The wheel’s faces will be perpendicular to the camera and will spin clockwise. Overlaying the wheel is a bracket that highlights each color. The purpose of the bracket is to show to the player what color will be shot if the “trigger” is pulled at that moment. At one time the player should be able to see the bracketed color and also the three following colors.
This will enable the user to strategize on who they will shoot for next.

Each color also has a corresponding symbol marked on the Chroma Cycle, which will allow for players with a Color-Blind deficit to be able to play our game.

The Chroma Cycle changes color every second and rests on that color for a second. The programming of this must allow for the ability to speed up and slow it down based on power-ups and difficulty involved.

4.4.4 :: Chaos Meter

The Chaos Meter is represented as a quarter circle that underlines the Chaos Cycle and the Chroma Cycle. It is located directly below these two items. This circle must resemble some form of stone architecture and accumulate circular “points” as the player racks up combos. Once the player accomplishes 12 of these points the meter is full and it glows white to notify the user that they may use the Chaos Cycle to select a power-up.

2-Hit Combo = 5%
3-Hit Combo = 15%
4-Hit Combo = 25%
5-Hit Combo = 50%

Combo Radius = 100 cm around each enemy

4.4.5 :: Chaos Cycle

The Chaos Cycle will be positioned stage left of the Chroma Cycle. It will also be half of its height and width and be centered against
the Chroma Cycle. The bracket which the Chroma Cycle uses also
has a sharp point on the bottom right-hand corner, which will
overlap the Chaos Meter. When the Chaos Meter is filled the user is
allowed to activate the Chaos Cycle and when they do, whatever is
highlighted by the point is the power-up they will receive.

The Chaos Cycle will spin twice as fast as the Chroma Cycle does.
This means that it will change from power-up to power-up in half a
second, which means that the player will have a harder time at
obtaining their desired power-up. The Chaos Cycle will also
include symbols, which correspond to different power-ups the
player can obtain once the Chaos Meter is filled.

4.4.6 :: Score

A total score is calculated as the user proceeds through each level.
The player can gain points by killing enemies and completing
combos. A more thorough breakdown of points can be found in
section “4.4.5: Point Distribution”.
4.5.0 :: Distinguishing Features

4.5.1 :: Chroma Blasting
Because Gentry drank the chroma chemical in the pool, he now has the power to blast out shots of chroma (color) infused energy from the palms of his hands. Each enemy is vulnerable to a different chroma and will be destroyed only when you use the chroma that is the same of the creature. If you hit an enemy with the right colored blast, enemies of the same color that are also in a small proximity to the damaged enemy are also vaporized. Hitting an enemy of a different chroma has no effect. After each shot, it takes roughly a second to charge the chroma force again. This cuts back on a rapid fire fest and makes the player pick and choose his battles. To illustrate:

4.5.2 :: Chroma Cycle
The source of Gentry’s power will be the color wheel, a meter which is constantly changing the chroma of your blast. Using this you will be able to tell what chroma you can fire and thusly what creature you will be able to destroy. There will be a static bracket around one spot on the wheel and whatever chroma is in the box determines which chroma you fire. To fire you simply press the left-mouse button and whatever is in the static bracket is the color that Gentry will shoot.

4.5.3 :: Chaos Cycle
The Chaos cycle is the second wheel, located next to the Chroma Cycle. This wheel constantly spins much like the Chroma Cycle, but its use is dependant on a meter. The said meter is charged by
racking up combo kills. When the meter fills, the player can then use the right mouse button to activate the power up. Which power up is used depends on which symbols is under the bracket when the right mouse is pushed.

**4.5.4 :: Available Power-ups**

Below is a list of power-ups that will be available to the user unless they are marked with an asterisk, which means that the player must unlock these power-ups.
For more information on how to unlock the noted power-ups please see the Point Distribution section below.

4.5.5 :: Point Distribution
The user may gain points based upon two different factors: how many enemies they kill and if they beat the par time for that level. The user’s score is displayed as part of the HUD and will be continuously changing based on his actions. At the end of each level the points are tallied up to inspect whether or not enough were acquired to gain unlockable power-ups.

The first point total is based off of how many kills the player makes. If the player has enough points in this total they will be able to acquire the unlockables of invincibility and clearing the room. These will be added to your Chaos Cycle and be able to be accessed during the following levels. A breakdown of these total points is depicted below:

![Kill Point Distribution Diagram]

- **Each Enemy Kill = 100 Points**
- **Enemy Combo Kills = (100 Points * (# of Enemies)^2)**
- **Defeating Level’s Boss = 5,000 Points**

---

**Kill Point Total**
The second point total is based off of whether or not you have beaten the par or average time for the level. If you have not there will be no tally of your points and you will not be able to achieve the unlockables. If you acquire enough of these time points you will be able to acquire the power-ups of rate of fire and scroll wheel Chroma Cycle control. These unlockables will be made available during the following levels. A breakdown of these time points is depicted below:

**TIME POINT DISTRIBUTION**

**LEVEL TIME POINTS**

$$\left(100 \times (\text{Par Time for Level}) - (\text{User's Time})\right)$$

**BOSS TIME POINTS**

$$\left(50 \times (\text{Par Time for Battle}) - (\text{User's Time})\right)$$

TIME POINT TOTAL
4.6.0 :: Controls
4.6.1 :: Control Diagram

Sake of Clarity’s younger target audience means that the controls for the game need to be very simplistic so not to confuse our users. Here is our control scheme as well as an explanation which follows:

The ESCAPE button will take you out of the game environment and bring you to a menu where you can save your game, load your game, quit the game and adjust game play components.

To move your character in 3D space you can utilize the W, A, S, D as well as the arrow keys.

The action button performs multiple functions depending on the situation. In most circumstances it will allow Gentry to jump, in others to dodge, and still others to push blocks around the game environment. The user may do this by pressing the SPACEBAR.

The last three buttons can be found on the mouse. Along with the movement keys the mouse enables the user to look around. It also allows the user to shoot with the LEFT MOUSE button, zoom in-and-out of your perspective with the MIDDLE MOUSE WHEEL, and utilize a power-up with your RIGHT MOUSE button.
5.0.0 :: NAVIGATION CHART

5.1.0 :: Start Menu

START MENU

NEW GAME
   EASY
       GAME STARTS
   NORMAL
       GAME STARTS
   HARD
       GAME STARTS

LOAD GAME
   SAVED GAMES SHOWCASED
       GAME STARTS

OPTIONS
   CONTROLS
       CONTROL OPTIONS DISPLAYED
   SOUND
       SOUND OPTIONS DISPLAYED
   VIDEO
       VIDEO OPTIONS DISPLAYED
   GENERAL
       GENERAL OPTIONS DISPLAYED

CREDITS
   DISPLAYS VIDEOS CREDITS OF OUR TEAM

QUIT
   QUILTS PROGRAM TO DESKTOP
5.2.0 :: Escape Menu

ESCAPE MENU

RESUME GAME
RESUMES GAME

LOAD GAME
SAVED GAMES SHOWCASED
GAME STARTS

SAVE GAME
ABILITY TO SAVE CURRENT GAME

OPTIONS

CONTROLS
CONTROL OPTIONS DISPLAYED

SOUND
SOUND OPTIONS DISPLAYED

VIDEO
VIDEO OPTIONS DISPLAYED

GENERAL
GENERAL OPTIONS DISPLAYED

QUIT
QUITS PROGRAM TO DESKTOP
5.3.0 :: Options Menu

OPTIONS MENU

CONTROLS
KEYBOARD CONFIGURATION
MOUSE SENSITIVITY
TARGETING RETICULE TOGGLE

SOUND
SOUND EFFECTS VOLUME
MUSIC VOLUME
DIALOGUE VOLUME

VIDEO
VIDEO RESOLUTIONS
GAMMA/BRIGHTNESS
CONTRAST

GENERAL
SUBTITLES

APPLY CHANGES
CANCEL

5.4.0 :: Level Movement

LEVEL MOVEMENT

MICHTALON’S TEMPLE
FOUNTAIN ROOM
PIT ROOM
TOWER ROOM
MIRROR ROOM
POLE ROOM
WOOWOO ROOM

TEMPLE ESCAPE
AMAZON JUNGLE
CAVE
DESERT VALLEY
BALBARINGER MOUNTAINS
6.0.0 :: SCENES AND ACTION

6.1.0 :: Rockimichili’s Temple
6.1.1 :: Fountain Room

<table>
<thead>
<tr>
<th>Object</th>
<th>Dimensions(cm)</th>
<th>Completed</th>
<th>Implemented</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base level</td>
<td>N/A</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Fountain</td>
<td>320x320x176</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Torches</td>
<td>80x80x115</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Door Locks</td>
<td>N/A</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Block</td>
<td>N/A</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Column</td>
<td>N/A</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Door</td>
<td>N/A</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
This is essentially a training room for the player to learn about the game environment and how to move and act in it. At the beginning of this room there is a cut scene which plays that shows Gentry drink from the fountain in the middle of the room. After drinking the strange multi-colored concoction the walls of the temple start to rumble and a small amount of enemies come out of the whole in the wall. Gentry then starts to go into what looks like convulsions but begins to shoot out different chroma from his hands. The cut scene then shows a number of the enemies getting hit by their corresponding chroma and dying. This will show that the user must match the colors in order to kill them. The cut scene ends and the player now only has a couple of enemies to deal with.

The enemies in this room are low in abundance, but will continue to come out as long as the player hasn’t blocked off the hole with the colored block. The player can do this by either shooting it or more accurately moving it with the action button. Once the player feels comfortable in this environment they can proceed to figure out the game’s first puzzle.

The puzzle in the Fountain Room is very simplistic and is just means to be built off of in future rooms/levels. The user simply has to shoot off two different colored locks on the door with their corresponding chroma. Once the user has done this the doors to the next room will open.
6.1.2 :: Pit Room

This next room is simply a tunnel, which will teach the user that certain environmental objects maybe used during the course of the game. The hallway is fairly long and as the player approaches the middle of it he notices that the floor has dropped away and there is a deep pit with spikes at the bottom of it. Once the user scans the world for a possible solution, they might figure out that there are strange looking rocks above where the pit is. If the user shoots them with any color of the Chroma Cycle a couple of times they will rumble, and finally after a few more shots, they will fall to the ground forming a clear path to the next door and the next room.

<table>
<thead>
<tr>
<th>Object</th>
<th>Dimensions(cm)</th>
<th>Completed</th>
<th>Implemented</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base level</td>
<td>N/A</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Boulders</td>
<td>N/A</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
6.1.3 :: Tower Room

The Tower Room is called so because of its immense height. In this room the user will find a strong enemy resistance, but more room to work and get around them if need be. There are a number of different enemy entrances that can be blocked off with a quick flick of the wrist by the user. Once they feel comfortable the user will have to combine different colored blocks in order to access higher levels in the room. The user will keep using this logic of jumping, building, and gunning until he gets to the last platform, which has a door that leads out of the room.
6.1.4 :: Mirror Room

The mirror room is the hardest puzzle of the game up to this point and Gentry will be in confined quarters with a strong flow of enemies to deal with. The user must first align the mirrors to hit the prism in the middle of the room. Once the mirrors are aligned in such a manner the user may choose to seal off the two enemy entrances. The user will then be able to fully focus his attention on the prism and mirror puzzle. Essentially, what the user must do is align the mirrors in such a way that the concentrated light beam blows the door open.
### 6.1.5 :: Pole Room

As the Mirror Room had the hardest puzzle, the Pole Room has the strongest concentration of enemy forces up to this point. There are four different enemy entrances, which flank each wall of the room. The user will have to be on his toes and be able to use strategy in picking off different enemies. Once the user feels comfortable he will need to hit the massive center pole 13 times before the gate to the final room opens.

<table>
<thead>
<tr>
<th>Object</th>
<th>Dimensions(cm)</th>
<th>Completed</th>
<th>Implemented</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base level</td>
<td>N/A</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Pole</td>
<td>N/A</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Door</td>
<td>N/A</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
The user enters this room and is at first in awe with the beautiful architecture that is all over this circular room. As he makes his way to the far side of the room he notices that there is a shrine built and right in the middle of this shrine sits the beautiful relic that is the WooWoo nut. As the player nears the WooWoo nut a huge rumbling shakes the temple to its foundation and a rock statue, which was sitting at the other end of the temple, awakens and makes a big commotion and he hovers into the center of the room. This is the great Merdunkian god of war, Rockimichili, and he is pissed.

The player must now face off against his antagonist and defeat him before he can steal back the WooWoo nut. Rockimichili has four different faces, each of these faces have two different attacks. One of the attacks will be able to be deflected back at him while the other attack will simply only be able to be avoided by hiding behind objects scattered around the room. The object in the attack that can be deflected back will involve a certain chroma. The player
must hit the projectile with the corresponding chroma in order to
hit Rockimichili. In order to defeat Rockimichili the player must
hit him with these deflected attacks 6 times.

Once the player has defeated Rockimichili, in a desperate, last-
ditched effort, will steal the WooWoo nut from you and will run
through the door as the player runs after him. This will be the start
of the next level, Tunnel Escape.

6.2.0 :: Temple Escape
This level will be time based as the player must be continuously
moving away from the temple, which is coming down all around.
The player’s mindset is two fold. The first is to try and catch
Rockimichili who has just stolen the WooWoo nut back. The user
will be able to see Rockimichili at all times in front of them. The
second thing is that the player needs to stay ahead of the rubble that
is falling down all around or else the user will be crushed by falling
debris.

This will not be easy because Rockimichili will throw numerous
amounts of enemies at the player. The user will either choice to
avoid these enemies or in more confined areas they will have no
choice but to fight them off and continue on. There will also be
environmental problems to deal with as well. Numerous amounts
of pits and crags will be placed in certain areas, so the player will
have to run and jump over them in order to continue on. There will
also be pointy stalagmites that jut up from the floor and will have to
be avoided by the player as it is in their path. Rockimichili might
also throw some kind of liquid on the ground and if the player steps
in it, it will cause the player to immediately slow down a substantial
amount.

If the player is successful in dodging these numerous attacks and
environmental obstacles he will eventually make it to the mouth of
the temple and see the vast jungle in front of them.

6.3.0 :: Amazon Jungle
The Amazon Jungle goes back to the original game play set forth by
the first level. However, the jungle will be less restrictive and will
feel much more open than the temple level. This is not to say that
the player is free to go everywhere they like. They will be confined
on a lesser basis by hedges, rocks, and trees so that they can be kept
on the path of the game play. The user will be able to go from area
to area and have to solve numerous puzzles and experience a
different assortment of creatures native to the jungles surrounding
the temple.
There will be more emphasis on environmental puzzles as the user might have to topple trees in order to cross rivers or other obstacles. The user might also have to find objects hidden in huts in the jungle that will allow them to pass to a different area of the jungle. Another puzzle might be that the user will have to figure out a way to set fire to the torches in order to gain access to the next area. The torches will have to be lit in a certain pattern, which can be found in a symbol or paper that can be found somewhere else in the level.

The enemies of this area will be distinctly different than the ones fought off in the temple. There might be huge, moving Tiki-like structures replacing the Stocreep. The Fistucko might be replaced by a tiger or some kind of other jungle feline. Aquacivi will give way to a crocodile. And the Siffler will take on the form of angry-looking jungle birds. Other possible enemies may include monkeys, boar, or elephant.

There will be a number of pitfalls that dot the landscape that will cause instantaneous death to Gentry if he doesn’t heed the warning signs of the thatched coverings. Other obstacles will be rapid-moving rivers that if the user falls in will cause the user to perish. There also may be various traps that will pull the user up into a tree, the user will then have to blast their way down in order to escape.
Eventually the user will come to a clearing, which, on the other side, is the entrance to the next level, the cave. However, as the user takes a couple steps forward a huge elephant stampedes forward. This will be the boss of this level and he will not be easily defeated. Certain areas of his body will glow a certain color at different times. The player will have to time his attacks in order to hit this area at the right time. Eventually the elephant will tumble over and the user will be able to get into the cave.

6.4.0 :: Cave
The makeup of the cave is one grandiose room, which is utterly crawling with enemies and traps. The user will be able to walk anywhere they want in the cave, but will have to use all parts of it in order to pass on to the next level. The level itself will be very dark and dank looking with only a few torches or other light source dotting the level. This will create a very ominous mood for the player to try and play through.

The puzzles of this level will take place in different areas of the level, but will eventually all lead to one thing, the collapse of the cave. The player will have to put together puzzles in 3 or 4 areas of the cave, which will essentially be rigging the cave to blow up. Once the user makes it to the final area and has setup the cave to blow they will have to fight the main boss.

The enemies of this level will all be animals that make their home in caves. For instance the Fistucko will give way to a rabid Raccoon. The Stocreep will give way to a massive, long-legged spider. The Siffler will give way to a pack of bats. And Aquacivi will become a rather large salamander. These animals seem to be glowing in the darkness, which will make it easier for the player to see them, however they do pop out of nowhere most of the time in order to disorient the user.

The final boss will be found in a rather large pond that can be found near the end of the level. It will essentially be like a large Plesiosaur, with a number of different heads or octopus like tentacles. There will be one enormous head that is constantly trying to take a bite out of you. The user will have to hit their tentacles with the corresponding color in order to take down this ancient dinosaur. Eventually the Plesiosaur explodes causing the cave to collapse and Gentry to make a narrow escape.

6.5.0 :: Desert Valley
After exiting the cave the user will find themselves in a desert valley, much like the steppes of the Southwestern United States.
Within this level the user will be confined by the huge walls that run parallel throughout the level. In this level the user will have to blast through rock and put objects together in order to proceed to the end of the canyon.

In this setting the user must fight off enemies such as vultures, scorpions, camels, jack rabbits, and lizards in order to reach the entrance to the pathway up the Balbaringer Mountains. Once the user reaches the end of the valley they will have to take on a Ferruginous Hawk that will take to the air and come swooping in on you as an attack. The user will have to use their targeting skill in order to take him down.

### 6.6.0 :: Balbaringer Mountains

The Balbaringer Mountains is the birthplace of the WooWoo nut and the user will have to venture into this unknown, icy, wilderness in order to complete his quest to figure out the Merdunkian society and more importantly, obtain the WooWoo nut. As the player continues to climb the steep incline to the summit the mountain itself starts attacking him. Rocks will tumble and need to be avoided. Trees come alive and try and squish the user. There will also be a wide variety of enemies in this level. For instance, the player will have to battle polar bear, mountain cougar, bighorn sheep, and ravens. The action will be furious throughout the level with a massive onslaught towards the end of the level.

The player quickly realizes that Rockimichili himself has been transformed into the mountain range, and is gunning to take the user down. If the user makes it to the summit they will have to fight Rockimichili once last time. Rockimichili will now be huge, nothing like what you battled in the first level. He will continue to use his different faced attacks in order to take you down. This time, however, his attacks are much more lethal, and come faster than before. The user must survive these attacks as the mountain starts to be eaten away by them. There will be very little room for the player to move around by the end of the battle. If the user is successful in taking down Rockimichili they will end the game in possession of the WooWoo nut.
7.0.0 :: RESOURCES

7.1.0 :: Models
  7.1.1 :: Gentry
7.1.2 :: Rockimichili
7.1.3 :: *Aquacivi*
7.1.4 :: Fistucko
7.1.5 :: Stocreep
7.1.6 :: *Siffler Swarm*
### 7.2.0 :: Sound Effects

#### 7.2.1 :: Aquacivi

<table>
<thead>
<tr>
<th>Sound Activation</th>
<th>Sound Effect Description</th>
<th>Sound Found?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack</td>
<td>Snake Hiss + Crunch of Flesh</td>
<td>Y</td>
</tr>
<tr>
<td>Walk</td>
<td>Slithery wet swishing</td>
<td>Y</td>
</tr>
<tr>
<td>Die</td>
<td>Water explosion</td>
<td>Y</td>
</tr>
</tbody>
</table>

#### 7.2.2 :: Fistucko

<table>
<thead>
<tr>
<th>Sound Activation</th>
<th>Sound Effect Description</th>
<th>Sound Found?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack</td>
<td>Snarling dog + Chomp + Crunch of Flesh</td>
<td>Y</td>
</tr>
<tr>
<td>Walk</td>
<td>Padded run</td>
<td>Y</td>
</tr>
<tr>
<td>Die</td>
<td>Firey explosion</td>
<td>Y</td>
</tr>
</tbody>
</table>

#### 7.2.3 :: Stocreep

<table>
<thead>
<tr>
<th>Sound Activation</th>
<th>Sound Effect Description</th>
<th>Sound Found?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack</td>
<td>Grunt + rocky smash</td>
<td>Y</td>
</tr>
<tr>
<td>Walk</td>
<td>Small rumblings with each step</td>
<td>Y</td>
</tr>
<tr>
<td>Die</td>
<td>Rocks tumbling + explosion</td>
<td>Y</td>
</tr>
</tbody>
</table>

#### 7.2.4 :: Siffler Swarm

<table>
<thead>
<tr>
<th>Sound Activation</th>
<th>Sound Effect Description</th>
<th>Sound Found?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack</td>
<td>Bee sound toned down + small knife like entrance</td>
<td>Y</td>
</tr>
<tr>
<td>Walk</td>
<td>Multiple Humming Birds in flight + locust</td>
<td>Y</td>
</tr>
<tr>
<td>Die</td>
<td>Buggy squish explosion</td>
<td>Y</td>
</tr>
</tbody>
</table>

#### 7.2.5 :: Gentry

<table>
<thead>
<tr>
<th>Sound Activation</th>
<th>Sound Effect Description</th>
<th>Sound Found?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reg. Attack</td>
<td>Like a zing-zing sound</td>
<td>Y</td>
</tr>
<tr>
<td>Power-up Attack</td>
<td>Deeper toned zing-zing</td>
<td>Y</td>
</tr>
<tr>
<td>Power-up Activation</td>
<td>Congo drums (bong-bong-bong)</td>
<td>Y</td>
</tr>
<tr>
<td>Walk</td>
<td>Two kinds: Stones / Sand</td>
<td>Y</td>
</tr>
<tr>
<td>Jump</td>
<td>Grunt + Harder landing</td>
<td>Y</td>
</tr>
<tr>
<td>Action</td>
<td>Sound Effect</td>
<td>Sound Found?</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Push</td>
<td>Extended Grunt</td>
<td>Y</td>
</tr>
<tr>
<td>Pull</td>
<td>Different kind of extended grunt</td>
<td>Y</td>
</tr>
<tr>
<td>Damage</td>
<td>“Blimey!” / “Ouch!” / “Get off!”</td>
<td>Y</td>
</tr>
<tr>
<td>Die</td>
<td>“Ughhhh”</td>
<td>Y</td>
</tr>
<tr>
<td>Idle</td>
<td>“Good thing I wore underwear today” / “Did I leave that burner on at my estate?” / “Blimey, can someone stop those horrible congo drums?”</td>
<td>Y</td>
</tr>
</tbody>
</table>

### 7.2.6 :: Environment

<table>
<thead>
<tr>
<th>Sound Activation</th>
<th>Sound Effect Description</th>
<th>Sound Found?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block Movement</td>
<td>Grinding against sand</td>
<td>Y</td>
</tr>
<tr>
<td>Doors Opening</td>
<td>Huge rumbling sound</td>
<td>Y</td>
</tr>
<tr>
<td>Fountain</td>
<td>Fountain sound- duh!</td>
<td>Y</td>
</tr>
<tr>
<td>Room Tone</td>
<td>Echoy draft</td>
<td>Y</td>
</tr>
<tr>
<td>Door Failure</td>
<td>Locked indicating sound</td>
<td>Y</td>
</tr>
<tr>
<td>Puzzle Completion</td>
<td>Bell ringing</td>
<td>Y</td>
</tr>
</tbody>
</table>

### 7.2.7 :: Music

<table>
<thead>
<tr>
<th>Room</th>
<th>Sound Effect Description</th>
<th>Sound Found?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pool Room</td>
<td>Calm tribal beats</td>
<td>Y</td>
</tr>
<tr>
<td>Pit Tunnel</td>
<td>Little more intense tribal beats</td>
<td>Y</td>
</tr>
<tr>
<td>Pillar Room</td>
<td>Most intense tribal beats</td>
<td>Y</td>
</tr>
</tbody>
</table>
7.3.0 :: Backgrounds

7.3.1 :: Main Menu

---

7.3.2 :: Loading
7.3.3 :: Death

![Image of a gravestone with the text: "GENTRY "A MAN OF A DIFFERENT COLOR".](image-url)
7.5.0 :: Milestones

7.5.1 :: Animation

11/17/05  75% (focus on Gentry-run, walk, jump, climb, push)
11/24/05  100% of character animations due (death, attack, movement) - put into GS
12/1/05   50% of environmental animations due (doors opening/closing, pool water)
12/8/05   100% of environmental animations due – put into GS
12/17/05  Anything left over

7.5.2 :: Modeling

11/17/05  Pool Room completed
11/24/05  Pool Room fully functioning in GS / Pit Hallway Completed
12/1/05   Pit Hallway fully functioning in GS / Pillar Room Completed
12/8/05   Pillar Room fully functioning in GS
12/17/05  Anything left over

7.5.3 :: Presentation

11/17/05  Milestones, audio list, script of first level, any other assets collected
11/24/05  Come in with all assets...
12/1/05   Come in with all assets...
12/8/05   Present final prototype and PowerPoint for the Professor
12/17/05  Present in large screening room the finale

7.5.4 :: Programming

11/17/05  Work on timed firing / killing of specific enemies / enemy AI
11/24/05  Implement pool room into GS / respawning enemies / get all characters into GS / add SFX / puzzle advancement out of room script / chroma cycle video
12/1/05   Implement Pit Hallway / puzzle advancement out of hallway script / enemy AI to a point where they behave decently / add SFX (music/environment)
12/8/05   Implement Pillar Room / puzzle advancement out of room script / anything else?
12/17/05  Anything left over

7.5.5 :: Menus/etc.
11/17/05  Nothing
11/24/05  Create menu graphics / implement a simple menu into the game / create GUI
12/1/05   Fully implement out-of-game menu / Add GUI into game (50% functionality)
12/8/05   Fully implement GUI into game / Create/implement in game menu
12/17/05  Anything left over

7.5.6 :: Audio
11/17/05  Audio List Completed
11/24/05  Add walking, attacking, dying sound effects for each character
12/1/05   Add environmental SFX – ambiance, mood music
12/8/05   Add Gentry comments
12/17/05  Anything left over

7.5.7 :: Documentation
11/17/05  Update with a section entitled Milestones, Audio List, Script
11/24/05  Fix all mistakes that were commented on the midterm, add any necessary sections
12/1/05   Start getting look down / add in game shots to graphics / start documenting problems and tutorials we had/used
12/8/05   Finish this sucker off
12/17/05  Anything left over
8.1.0 :: Programming

8.1.1 :: Action

Unlike C++, GameStudio has more than one type of function. Both of GameStudio’s types are nearly identical in format and function, but are used for different purposes more as an organizational means. The new type of function GameStudio has is an Action.

An action is the same thing as a function however you apply it to a piece of geometry; this allows you to give the piece of geometry instructions. These instructions are often done inside while loops that execute every frame. By doing this you can make a piece of geometry move every frame based on criteria or even give it complex AI to react in a multitude of ways.

```
action move_me
{
    While(1)
    {
        Wait(1)
    }
}
```

Above shows you the basic format of an Action with a while loop that executes every frame. Telling the while loop to wait(1) makes so that it only executes once a frame. If you didn’t do this your loop could go on forever causing your computer to crash.

For Chroma Chaos we use Actions for our main character as well as each monster, and each interactive object. Each one of these has their own action so that we can give each one different “personalities”. Although it would be more efficient (in terms of lines of code) to create one monster or object action and pass different information based on the type to a function, it would require a vast revamp of our current codes as well as possibly be harder to navigate. Also seeing as the game we are set out to make isn’t particularly large at this point in time we don’t foresee and issues with our current program layout.

8.1.2 :: Function

I won’t even go into what a function is, because I assume you know. As with C++ there is a main() function which initiates everything. You’ll notice that there isn’t much of anything in ours. Even in
terms of functions it calls. This is mainly because Actions and geometry interactions are a large portion of our game thus most of the code is contained within these. Also due to Game Studio’s level creator, most static objects as well as the map itself are placed there opposed to in the code. It is possible to place models from within the code, you can see we place all models with actions within it, but for design purposes we found it easier to place them manually.

8.1.3 :: Organization
If you just try to scroll through our code you’ll realize most functions and actions are randomly placed. For the most part it really doesn’t matter where most of them are in the code. This is because we don’t use many functions or actions that aren’t self reliant. If they do need information from somewhere else it is often transferred by global variables. You can most likely tell this by the large amount of variables defined at the beginning of the code. I’m sure there are better ways to organize and transfer data but for our purposes we needed a quick and unfortunately dirty method of getting the game running. Luckily in the GameStudio Script editor there are code jumpers on the right hand side that will jump you to functions or actions you wish to view. Because of this we tried to give our functions and actions informative names so as long as you knew the name or the function of what you were looking for you could easily find it.

8.1.4 :: Time
Throughout the code you’ll see “time” being used in movement distances or shot timers and etc. This is because the game does not run at a static frame rate. If the code is being executed every frame then the game play would speed up or slow down based on the frame rate. To make up for this the movement values and others have “time” incorporated into them. “Time” is something Game Studio has built into it, it keeps track of how much real world time is spent on each frame. If there are more frames then “time” is smaller, thus movement multiplied by time will be smaller. This makes sure everything moves/executes at the same rate regardless of frame rate.

8.1.5 :: Player
Explained in the actions section the player uses a constantly running while loop to execute all actions associated with it. It is constantly looking for the player to press A, S, D, W, SPACE or MOUSELEFT. These are detected using an if(Is pressed). If the key is pressed code is activated. You’ll see that the players location is kept track of by a global variable this will become important later. Also, when a player fires a shot the shot is an actual piece of
geometry with its own action. What chroma the shot is, is actually determined here and stored globally.

8.1.6 :: Fire
Fire is a relatively simple action. It causes an object to move away from the player’s location for a certain period of time. Then it destroys itself, all the while keeping track of its location globally. The shot action itself does not destroy or even have any awareness of anything around it. It simply goes forward and is more used as a tracking device rather then an actual destructive projectile (in terms of code).

8.1.7 :: Enemies
These were the most challenging part to code, mainly because even simple AI isn’t very simple. The basic purpose of the enemy is to move towards the player and attack them at a certain distance. This may seem easy but there is a relatively complex mathematical equation used to compare the enemy’s locations to the player, rotate the enemy so it’s facing the player, and move the enemy to the player. I won’t get into the math of it but just trust us when we say it works. On top of having to do all this, the enemy is constantly checking to see if a shot has hit it and if that shot is the same color. If both are true then the enemy destroys itself. Mokia box collision is also handled by the monster and it will move itself if it gets to close. All of the above is the main reason why so many object locations are stored globally.

8.1.8 :: What knows where what is and why?
This will give you a little more detail as to the reason and relation of our global variables. When designing the enemy code we came upon a conceptual wall as to how we would keep track of the enemies. We initially were going to make a vector of enemy locations however vectors in GameStudio don’t work the same as normal C++. Vectors are essentially simple arrays and don’t allow much manipulation. Thus we needed another way to keep track of everything. So our solution was who needed to know where what was.

The enemies are the only thing that knows where they are in the code. This eliminates the problem of keeping track of so many enemies. However because of this the enemies must know where everything else is so that they can react to other things inside the enemy. So the players location, shots fired and other interact able objects have locations that we store globally. This makes the code inside each enemy rather beefy but other objects code rather slim. Luckily the player doesn’t need to know where anything is inside the code. This is because the human player uses optical information to determine where the enemy is. Although all this makes where
interaction take place somewhat confusion it solves the problem of storing rather large amounts of data. So when in doubt look inside an enemy for an interaction because it most likely won’t happen in other areas of the code.

8.1.9 :: Particles
Each of our particle effects consists of one action, two functions and a bitmap, the graphical representation of the particle. The action gets applied to an invisible and passable object where the particles are to be emitted, the two functions cycle through each other changing the attributes of the particle effect as desired. Our main use of particles is in the enemies to create a color full explosion upon death. In order to achieve the explosions we have up to 12 particle systems spewing out particles every enemy death. It may seem like a lot but they are just still images being moved in space so it doesn’t put much strain on the computer. The other reason for having so many emitters is to create the AOE combo death, having particles spew in all directions away from the enemies.

8.1.10 :: Score
The score is displayed the same as the GUI, simply sticking an image up on the screen. We decided to do it this way to allow us to create fancy custom numbers to be displayed instead of a plain text display. As the player kills enemies a counter variable is increased and is mod divided to find out what number is in each place. Once we get the number in each place we call the appropriate image. This unfortunately causes us to have a very large amount of images due to our score being able to go well past 1000.(10 images are needed for a score of 0-9) At the same time a large amount of if statements must be run to figure out which image should be made visible.

8.1.11 :: Camera
The camera has gone through many different stages throughout our project and is still not perfect. The main problem being parenting it to the player to keep a constant 3rd person view doesn’t seem to work. If it does we’re just doing something wrong. So, instead, we created our own equations to base the cameras location off the players and rotate it as the mouse moves over the screen. This may seem simple but the camera doesn’t seem to be an entity and can’t be moved with ent_move. So in order to move it we have to give it coordinates to move to. We do this by finding the amount the player moves in their X and Y then moving the camera by this amount. This all works fine until the mouse is moved quickly. When this is done the camera becomes off center and is focused in the wrong area. We have not been able to figure out why but we suspect some sort of data being lost due to the player moving the mouse extreme distances over shorts amounts of frames.
8.1.12 :: Music
We are using the built in media loop to play music in the game. In order to switch music we use the media_stop that stops all the audio being played and then we start a new track. Unfortunately, this causes a significant pause in the game due the new music being loaded. It was suggested to us we have all our sounds playing at 0 volume from the start of the game then turn the volume up, down or pause as needed so that we wouldn’t have the loading issues. That seems like it would work however it still needs to be tested to see if it would work.

8.1.13 :: Sound Effects
Much like the music we load the sounds when needed however was not looped through media loop. Timer variables are used with if statements to make sure the sound won’t play until after it has finished its cycle once. This occasionally causes minor lag but the sound effects file are so small it isn’t noticed on a faster machine. All monsters and the player have sound effects on walk, attack and die. With the large amount of creators on the screen this would cause a large amount of sound to be playing at once. So in order to stop this equation was made to lower the enemy sound volume the further away they are. This also helps the player know if a monster is coming up behind them. Very useful seeing as we don’t have a mini map. The equation we use is:

\[48000/(|me.x-player\textunderscore location.x|+|me.y-player\textunderscore location.y|)\]

This is put in the enemy volume control to scale it up and down. The various other sounds on doors and such are simply played once when ever the player interacts with them.

8.1.14 :: Chroma Cycle
The Chroma Wheel was implemented rather strangely with a mix of code and animation to get it working properly. The spinning of the wheel is actually done through playing its animation at the proper rate. It was done this way because the wheel’s pivot point is directly where the camera is. If we were to rotate the actual geometry the wheel would spin around the player and not in place like we wanted. The pivot point was placed there so that when the mouse moves and pivots the camera the wheel pivots in the same way and stays with the camera in the same place. Since we were pivoting the wheel like the camera we decided to move it like the camera too by giving it exact coordinates. This was easy because we simply replaced its X and Y position with the cameras X and Y position anytime it moved. The colors that are on the wheel were placed in the texture editor to match the cycle we made in code. This way
nothing additional needed to be done to make the two line up. In code the wheel itself actually has nothing to do with what color is fired. It is simply a visual representation of it for the player.

8.1.15 :: Health Bar
This is moved the same way as the Chroma Wheel. However its animation was a little trickier. The bar has an idle animation, but after the player takes a hit and we tell the bar to play the first mouth closing animation we can’t have it play the same idle animation. We actually have four idle animations for the four health states with the mouth at the appropriate level of being closed. Which animation to be played is based on the player’s current hit points.

8.1.16 :: Chaos Cycle
We actually implemented this in a very basic manner. If a creature is killed from an AOE hit the chaos_ticker variable is increased by one. Like the score depending upon the number of the chaos ticker different GUI pictures are made visible to show this. When the chaos meter is full music plays and the player can use the clear screen power up by clicking however the chaos cycle does not spin and there are no options to choose from.

8.1.17 :: Intractable Objects
The intractable objects were placed in the map editor and assigned their actions in the map editor. For the most part whether or not their actions should accrue is based on shot location to their location comparisons. If the shot is within a certain range and for some is a certain color then do this. The animations are looped through like everything else and the sounds are usually just played once. There is noting particularly complex about any of these, but some of them are used as triggers for major music changes.

8.1.18 :: Mokia Box
This object is relatively complex compared to most of the other objects. In order for the player to move it by shooting it is compares the players shot location to it’s to determine if it’s hit. At the same time it is storing the last frames shot location to the current one. The difference in the two is the X and Y it should move. The player also has the ability to move the block (although relatively useless as currently implemented). When the player presses space it compares its location to the players to determine which direction the player is coming from. Then if the player is pressing back the block moves back with the player as a pull command, if the player moves forward so does the block. There is one last function of the block that does not actually accrue in the block, it accrues in the creatures. In every creature they are comparing themselves to the
blocks location. If they are within a certain range of the block they are told to be moved back to avoid colliding with the block. It’s not the most efficient way of handling this feature but it gets the job done.

### 8.1.19 :: Animations

When a criterion is met the animation is begun. The animation percentage is what is used to go through the actual animation. At 100 the animation has gone through one cycle, at 101 the animation has started again and can continue to go up infinitely. Holder variables update themselves every frame to add to themselves and progress the animation. When an animation shouldn’t be continued the variable is told to stop updating and running the animation command. Each model can have more than one animation attached to it. When exported each set of frames is given a name. These names can be referred to in code. If I were to refer to Gentry’s walk cycle at 100 percent it would play all the frames in his walk cycle but not all the frames of animation the model has.

### 8.1.20 :: Creature Destruction (Wrath of God)

Once we finished the first room we realized that we needed someway to destroy all monsters at once when the player left that room. Also when we decided to implement the chaos cycle we realized we needed this too. So in every creature there is a little if statement that checks for one variable. If this variable is set to the right number all the creatures will destroy themselves. As far as room to room these numbers are set when the player passes a certain world coordinate location. For the chaos cycle we simply tell the variable to be equal to the one needed when the player right clicks.

### 8.1.21 :: Gravity?

So we discovered that Gentry and monsters can walk over holes and not fall. We are not sure why this is because we thought it was built in. Apparently it isn’t. So in order to remedy this in our pit fall area, when the player crosses a certain world coordinate location the player’s model is told to move in the Z coordinate for a period of time then the death GUI is brought up. Rather crude but it simulates falling. If the boulders are down the code simply skips the falling part. Eventually a jump and gravity like fall will be needed. But for the current game demo it wasn’t needed extensively so this little hack was good enough.
8.1.22 :: Enemy Turning AI

Version. \( \alpha \)

\[
x_{\text{diff}} = \text{me}.x - \text{player}_\text{location}.x;
\]
\[
y_{\text{diff}} = \text{me}.y - \text{player}_\text{location}.y;
\]

The above equation determines the amount of distance from the enemy the player is. This is integral to finding the angle at which the enemy must turn to face the player.

\[
\text{if } (y_{\text{diff}} > 0) \\
\text{ if (} y_{\text{diff}} > 0 \text{)} \\
\{ \\
turn\text{deg} = \text{acos}(x_{\text{diff}}/(-\sqrt{\text{abs}(x_{\text{diff}} * x_{\text{diff}} + y_{\text{diff}} * y_{\text{diff}})})); \\
\}
\text{ else } \\
\{ \\
turn\text{deg} = -\text{acos}(x_{\text{diff}}/(-\sqrt{\text{abs}(x_{\text{diff}} * x_{\text{diff}} + y_{\text{diff}} * y_{\text{diff}})})); \\
\}
\]

Here, notice there is 2 equations; the reason for this equation is that all the angles up to 180 are negative, which is essentially the same thing. What is being calculated is the adjacent angle to hypotenuse. Since we have the length of the opposite and adjacent sides, we can use the Pythagorean Theorem to find the hypotenuse. So basically the above equation is:

\[
\text{turndeg} = \text{arccosine} \left( \frac{\text{length of adjacent side}}{\text{hypotenuse}} \right)
\]

\( (\text{length of adjacent side}) / \text{hypotenuse} \), gives you the length of the side opposite the hypotenuse. Since you have the side now, if you use a little trigonometry by taking the opposite of the adjacent side you will get the angle of the adjacent side.

That angle is the amount the enemy should turn to face the player. Which is:

\[
\text{c_rotate}(\text{me}, \text{vector}((\text{enemy}_\text{degree} - \text{turndeg}), 0, 0), \text{IGNORE\_YOU});
\]
\[
\text{enemy}_\text{degree} = \text{turndeg};
\]

By setting \text{enemy}_\text{degree} to turn degree we ensure that the next time the player moves the above equation factors in that the angle is different and calculates the angle from where you were facing and not at 0, of which \text{enemy}_\text{degree} is initially set to.
Finally to all those who care about this: **SOHCAHTOA**!

### 8.1.23 :: Perfect Circular Angle Theory

The following is only a theory due to the limited amount of brain power logic has, which in the end, the class much sleep to be lost by the writer. In theory it works, but to make it work in C is the challenge, naturally, if you wish to take part in it.

First we have the camera moving in a diamond shape, we know the radius is constant, since the camera is only a set distance away. We also know the angle of which the player turns. Since that is known the formula for finding the length of a circular arc can be found (note: *angle θ is the angle that the player turns*):

- **if the angle θ is in degrees**, then \( \text{length} = \theta \times (\pi/180) \times \text{radius} \)
- **if the angle θ is in radians**, then \( \text{length} = \text{radius} \times \theta \)

If we divide the answer by the *angle θ*, a.k.a. the angle at which the player turns. We get the amount of movement per degree. This doesn’t help much except if you want to calculate the angle at which the camera clips and object so that you can do something it avoid clipping.

Moving on since we have length of the circular arc we may calculate the distance the camera moves to get from the initial point to the target point so that the camera is still facing the player. Here is where things start to get messy...since we can get the hypotenuse by dividing by PI, from there we can find the angle adjacent to the hypotenuse by adding the angle that the player turns to 90 and subtracting that from 180 degrees.

This will give us the adjacent angle. Since we have that we then subtract that angle from 90 and get the angle to rotate the camera to orient it so that the camera is again facing the player.

To save the hassle of calculating interior, exterior, corresponding angle and such, it is recommended that for every 90 degrees the angle is reset to 0 degrees.
Here is a picture to help you visualize:

The gray circle is the length of the circular path.
The blue line is the length of a circular arc.
The green line is the direction the player will be facing with no correction.
The yellow line is the radius.

8.1.23 :: Useful Information
1. When you apply an action to a piece of geometry the game automatically stores its x, y, z pan, tilt and roll. You can access these values by using me.x, me.pan ect...
2. c_rotate() rotates a object by a certain amount of degrees not to the degree inputted( if you are facing 30 and you c_rotate 30 you will be at 60)
3. var player_location[3]; variables are like arrays. A variable of [3] is like an array of 3 data spaces. They can be accessed by using the variable name .x, .y or .z or using .pan, .tilt or .roll. This is because it is a game engine, and you are often working with object movement and rotation
4. ent_remove (me); destroys an object but if you do this inside a while loop make sure you do break; to stop the loop... how many hours we wasted on yeee...
5. ent_create( player_to_create, player_location, move_me); basic way of creating something (name of model, location of model, action)
6. ent_move(vector(0, -1, 0), nullvector); amount to be moved. Don’t know why but you need that nullvector part.

8.1.24 :: Resources
http://www.coniserver.net/ubbthreads/ubbthreads.php
http://s91902847.onlinehome.us/
8.2.0 :: Importing Animation into GS

8.2.1 :: Max to Maya
The best way to bring an animation from Max to GS is through Maya. After getting it into Maya you must then follow the steps outline in 8.3.2. There are two ways to convert a Max file into Maya. We have noticed that each one of these options have different difficulties, depending on the kind of animation, once they have arrived in Maya, however, this is the only way with GameStudio’s restrictions.

The first option is to export the file as a .3DS, an old Max format, and then import that into MotionBuilder. Save it out as an .fbx and bring it into Maya. This option has some problems with recognizing bones as bones but is better for SplineIK and grouped objects.

The second option is to export it as an .fbx and bring it straight into Maya, this can cause some objects difficulties in retaining their animation. After you have figured out a successful way to acquire your animation, follow the steps below to get it into GameStudio.

8.2.2 :: From Maya
To export characters with animations from Maya into GameStudio you need a MEL script for Maya called mdl5_export_v5. It can be found on Perforce under the “Game” folder in zip format.

In the Maya file a few things have to be done to the model before it can be converted. First, it needs to be triangulated, to do this select all of the parts of the model and go to Polygons->Triangulate. This has to be done, because GameStudio does not allow for models to be without triangulation of their faces. The next step is to setup the object’s textures so it can be properly mapped in GameStudio. To set this up you need to go to Edit Polygons->Textures->Automatic Mapping. This only works with objects that only have one base texture. If you are wondering how to properly skin a multiple textured object please refer to 8.4.0.

Once that is done your model is ready to export. Open up Script Editor and go to File->Open Script and select the mdl5_exporter from the destination where you unzipped it to, the script will appear in the window, select all of it and run it.

A window will open with a list of animations which you can add more to or remove from. Add a new animation for each animation the character will do in the game; you should already have them all on the timeline in this file. Then, set the frames for each using the format 1 2 3 4 ..... ##. Set the path to where you unzipped the
mdl5_export. A model.txt file will be created. Run the mdl5_convert_v4.exe file that was in the zip and a model.mdl file will be created that you can rename and use.

### 8.2.3 :: Triggering the Animations in GS

A few lines of code are required to use an animation in GameStudio. First a var needs to be created and set to 0 outside of the characters while loop for example “var ani_walk = 0;” then inside the while loop 2 lines “ani_walk (or whatever you named it) = ani_walk + x * time” where x will control how quickly it will play back. Then “ent_animate(me, "walk", ani_walk, ANM_cycle);”. The “me” designates the character whose action the code is in, the "walk" is what the animation was called when it was exported out from Maya, the “ani_walk” is that variable again and “ANM_cycle” will make it continuously play which is good for run cycles and so on. Another option for the “ANM_cycle” is “ANM_add” which will add the new placement to the existing one. This is good for mixing animations.
8.3.0 :: External Object Texturing into GS
Character Modeling through the Alias family of software:

Some characters were built in Maya, which had an excellent plug-in for exporting animated characters as MDLs, the modeling format used by GameStudio. This plug-in extracted each frame you specified to create sequences for specific movements. We also discovered this format saves the UV layout for the characters, allowing us to forgo using GameStudio’s weak UV editor, and instead doing it in Maya if we wish.

“Fistucko and Gentry” – Both of these characters began with a series of boxes, which were built on and expanded until the mesh, the characters structure, was complete. This system is time consuming, but great for game modeling as it gives a great amount of control to the modeler. Game models need to skimp on the amount of polygons used on them, since to many will slow the machine down, so the less, the better.

After modeling, the character then must be readied for animation. This takes two steps.

First, these characters had a skeletal structure built for them using Maya’s bones system. This would allow the animator to deform the skin of the character to make it walk, attack, and play dead.

Then, the bones must be rigged, a process which relates the bones to each other and controls how they bend. Using rigging, you can quickly and easily pose the character, so that moving their wrist will bend their elbow, or rotating their ankle will move the hip. This is done through Kinematics, both Inverse (IK), and Forward (FK). For this, Alias’ MotionBuilder was used, as it can quickly and easily set up the rig if specific naming conventions are used on the bone structure. Unfortunately, MotionBuilder is exceptionally buggy, and needed to be upgraded in order to fix an issue with the Fistucko’s legs bending incorrectly. The “effecter”, the control for the shoulder, was being placed on the elbow, even if named and linked correctly. Upgrading to MotionBuilder 7 fixed the issue.
8.4.0 :: Texturing w/GS

In order to obtain a textured model inside GameStudio, you must use GameStudio’s built-in model editor. Open the .mdl file in the editor that was created using the mdl exporter script in Maya. The model appears blue and untextured, so go to View->Skins. This opens another window that displays the model and the UV texture map. Any texturing that was done in Max or Maya is not saved with the mdl when exporting. If any type of UV mapping was applied in Maya, such as a cylindrical map, this is saved with the exported mdl and should be displayed. Different UV mapping can also be created in the skin editor by going to Edit->Create MDL mapping, but this will only create front, right, and top UV maps.

In the Skins window, go to File->Export Skin as a BMP. You can now open the bmp in Photoshop and paint over the wireframe of the object however you like. Be sure to completely paint over the white lines, as not doing so will display them on the model in the game.

After you have completed the creation of a texture in Photoshop, save the file out as a bmp. Make sure that the file has the same resolution as the skin image in the model editor, as it may not match up with the model when you bring the image back in. Go back to the skin editor and import the new bmp you have just created for your model. The model is now textured in the model editor, and you can save it as an mdl7 file to be used in GameStudio. The texture as well as the model are saved in the mdl file, and can be brought into GameStudio by going to Object->Add Model.

For texturing the map file within GS make sure you texture has one of the following:

-JPEG, BMP, PCX, TGA. Photoshop supports all 4.

It's recommended that the resolution be at:

-1024x1024, 512x512, 256x256, or 128x128 pixels.
-You may make a higher (not over 1024x1024) or lower resolution, but for best results make sure they are at a 1:1 ratio.

To smack a texture on an object GS...

On the left side of the in the WED (in Level Editor), click on the Textures tab.

Click on default.wad and right click in the now open field to open
the texture manager.

Then you can either click Add WAD and select the standard.wad, or Click New WAD and enter a name for your texture collection (like MyFirstTextures.wad).

Click OK.

Now you have either the standard textures (default.wad) (careful with the animated ones, they are tricky to do and will not be discussed in this tutorial) set in a box, which you can scroll down, or a blank black box for your own textures.

If you want to add your own textures, right-click in the black box and select Add Texture from the menu. Your texture dimensions must be multiples of 16, maximum size is 1024x1024, and in the format of PCX, BMP or TGA.

Select the object in the viewport in the level editor, the object should turn red when selected, and then double-click on your texture on the left.

The texture will appear on the top underneath the Object box. Now you have textured the object! Kudos to you, you may now get a cookie form the cookie jar.

Squirrel nuts!
8.5.0 :: Sound Effects
8.5.1 :: Sources
There were primarily four different sources from which I acquired sounds. Lacking the means of renting equipment to record sounds outside of the MUSC lab, I was forced to find everything I could in music libraries and internet websites. The music libraries proved to have quality sound effects, but lacked in the sounds that I wanted. The internet was the exact opposite. I could find all the sounds I wanted but it either was poor quality or cost money.

The first source was recording sounds myself. This was the most dynamic and fulfilling of all sources, but required the most work. There also weren’t many sound effects that I needed to record myself, because all the other sound effects were from environmental objects. Thus, the only sound effects I recorded was voice-over for Gentry. This was done in the MUSC lab with a dynamic mic.

The second source was the internet. Primarily, I used Flashkit.com to find pretty high quality sound effects. A number of sounds, however, could not be found on their database. If this was the case I simply did a google search for the specific sound effect. The problem with this is twofold. Usually the sound is really crappy quality or you need to pay for it.

The third source was Drexel’s Sound Library, which, despite its lack of pretty basic sounds, can be useful. The biggest problem with the library is that there isn’t a way to browse through the different categories of the library. In order to use it effectively you have to break out the thesaurus and try and find something by typing in similar words over and over. Even though the sounds in this library are mp3s, they are still very high quality, especially for a game.

8.5.2 :: Creation
The program I used to edit, mix, and bounce the audio files I collected was Adobe Audition. I find that the interface within this program is simple enough for anyone to use, but has the capability for an advanced approach as well. Each mp3 that was made from Audition were 16 bit, 44.1 kHz. I felt that this was a good quality sound and wasn’t going to totally bomb the system’s memory when playing the game.

Some of the game’s sounds such as the doors slamming shut were a combination of two different sound effects mixed together to form one complete sound. In all the sound effects I also tried to get the most gain out of them before they distorted so we have the best quality sound level to work with in game.
8.6.0 :: HUD

8.6.1 :: Requirements
The requirements for the HUD are relatively simple in GameStudio. The engine supports most popular resolutions, such as 800x600 and 1024x768. Right now our game is played in 800x600 but I was hoping to expand the game’s resolution to 1024. For this reason I always make two different copies of our HUD, one hi-res and the other low-res. This way when we come back and want to bump up the resolution a little we will already have the files on hand.

The other requirement is that whatever is black in your HUD, will show up transparent inside the game. This creates problems later on, but can be avoided if careful planning is used.

The last requirement is that GameStudio only accepts bitmap images for pretty much anything graphical in the game. The HUD is no different, it needs to be saved out as a bitmap.

8.6.2 :: Techniques
For making our HUD I used Photoshop CS. I opened up a 1024x768 composition and began designing away. From Pat’s art I had a pretty good idea of what he wanted and used the standard convention of items in the corners of the HUD so that the vision of the user is never impaired. This was easy to pull off because there aren't many times in our game that the user will have to shoot something vertically.

The hardest thing about creating a HUD was not getting little black speckles where once a drop shadow or some other black color was. This is because GameStudio tries to create an alpha channel wherever there is black involved. To avoid this try using other colors as drop shadows or glows. You could also use colors that are close to black but are not in fact black. This is what I used for the different drop shadows that are behind each object of our HUD.

8.6.3 :: Layering the HUD
The game called for a portion of our HUD to be in a constant, looping motion. Because Photoshop cannot do this we decided to render this out of Maya as an animation video and just layer the HUD over this. In order for this to all fit together perfectly, we first applied the video layer in our HUD and moved it around to find the best position for it. Once this was found, we took a screen capture of our game. I then brought this into Photoshop and cut out the shape of the video layer and used this selection to build around it, making sure not to over or underestimate the video layers size.
Once the HUD was fully laid out in Photoshop I saved it out as a bitmap and brought it into GameStudio where it was layered over the video.
8.7.0 :: Journals
8.7.1 :: Matt Parent

10.16.05
-Downloaded/Checked out Game Studio

10.22.05
-Started initial design document layout

10.23.05
-Continued work on design documents
-Started to work on Stocreep
-Finished work on Stocreep

10.24.05
-Continued work on design document (section 1-3.3.2/build 1.0.0.1)

10.27.05
-Continued work on design document (section 3.3.2-4.5.0/build 1.0.0.2)

10.29.05
-Continued work on design document

10.30.05
-Continued work on design document

11.2.05
-Did around 25+ pages more of design document plus all the graphics (build 1.0.0.4)

11.12.05
-Sat in and analyzed our current game code
-Helped Alex/Matt with the programming

11.14.05
-Made the audio list

11.15.05
-Made the milestones list

11.19.05
-Started to collect music for our game prototype

11.20.05
-Finished the Torch model.

11.21.05
-Finished the fountain model.

11.25.05
-Released build 1.0.0.6 of our Design Doc- with fixed sections

11.26.05
-Made improvements to GUI
-Made and textured a column object

11.27.05
-Added walk, death, attack sounds for Fistucko, Aquacivi

12.4.05
-Constructed an Alpha Version of the HUD
-Constructed a Loading Screen
-Added sounds for all of Stocreep,

12.6.05
-Constructed a death screen
-Put together the PowerPoint Presentation

12.7.05
-Constructed the score portion of the HUD
-Added new gentry walk sound
-Added new boulder falling sound (for stalagmites)
-Added new converted door dropping sound
-Added new HUD w/new reticule position
-Added new power-up dots for the Chaos Cycle
-Worked and released version 1.0.0.9 of our Design Document
-Recorded Gentry's sounds
-Practiced with group mates

12.9.05
-Finished the brochure for our presentation
-Worked on the DD a lot. Added graphics/a few journals

12.10.05
-Wrote more about other regions of the game in the DD
-Made a new power-up picture for the DD
-Added Matt and Josh's journals to the DD
-Colored/Labeled the Regions Graphic

12.11.05
-Added the rest of the journals/appendixes to the DD
-Colored the Game Art Graphic
12.12.05
-Made a combo graphic
-Fixed the combo graphic/combo ticks
-Made four different rugs for our level

12.13.05
-Met with team assigned roles and practiced for presentation
-Finished editing the DD
-Recorded Gentry audio for intro movie.

12.14.05
-Met with team again and practiced for presentation
-Binded DD

8.7.2 :: Matt Smith
Week 1
I came up with original idea for a color-based shooter. Then, I reworked game with teammates to be more interactive. I also thought up the enemy concepts.

Week 2
I spent this week thinking up and planning out the animations that we would need for the game and, of course, refining our game again. Wrote up Boss fight animations/ game play.

Week 3
This week I started out by waiting for the modelers to get me things to animate, but the wait was not long for Josh and Ari came through with a dog and a snake respectively. I spent the week end animating those, as well as Sunday night and then again on Monday working with Alex to try to get them into GameStudio. On Tuesday I worked with Pat on fixing some of the modeling (and mental) issues he was having with Rockimichili.

Week 4
Worked on programming with Alex, did more animating, started animating Gentry.

Week 5
Made animation of puzzle example for midterm, animated more actions of Gentry. I also animated and helped Alex with getting the enemy AI working.

Week 6
Started working on trying to get animations from Max-to-Maya-to-GameStudio. More animating of Gentry. More animating of other enemies. Continued working on code and got basic concept of AI.
Also worked on shooting with colors.

**Week 7**
Finished animating Gentry and converted him over to GameStudio to be implemented. Implemented code for multiply enemies. This broke the code for the enemy AI so we started working on finding a solution to that. Started re-animating redone enemies.

**Week 8**
Finished Re-animating enemies, Figured out how to get enemies from Max-to-Maya through MotionBuilder and then on into GameStudio. Worked on code for camera and finished fixing the AI. I wrote up section on conversion of enemies from Max-to-Maya-to-GameStudio.

**Week 9**
Got all of the Enemies into MotionBuilder with animations and coded the doors, spawning of enemies in other room. Added sounds in game. Got the falling down the hole to work. Got the falling boulders to work and buttons to work. clearing the room after the player leaves.

**Week 10**
Made an opening video for our game, coded the point system and I also coded our power up and chaos meter and our puzzles. The pillar in the other room getting to go up and hitting the roof, causing the door to Rockimichili to open and all the enemies to die.

**8.7.3 :: Ari Teger**

**10/20/05**
-Began modeling the low-poly beetle and most of the Aquacivi serpent

**10/21/05**
-Finished modeling the Aquacivi
-Animated the wings for the Siffler Swarm

**10/26/05**
-Animated flying paths for the Siffler Swarm animation
-Modeled a boulder and torch

**10/30/05**
-Animated the attack animation for Siffler Swarm
-Worked more on the menu design in Photoshop

**11/2/05**
-Rendered out animations of the Aquacivi and Siffler Swarm to be
used in the presentation on Thursday
-Finished the menu screen

11/9/05
-Began working on the level in Max

11/12/05
-Worked on level design more

11/15/05
-Revamped Siffler swarm models to make them look more menacing/changed the wings
-Changed the fly animation for Siffler swarm

11/16/05
-Finished designing the prototype level in Max

11/20/05
-Created a puzzle switch model for the first room

11/26/05
-Started creating the pillar for the pillar puzzle room
-Created and textured the puzzle switch

11/27/05
-Finished the rising pillar model and textured it

11/29/05
-Re-modeled parts of the level, with added holes and rooms for enemies to spawn in

12/3/05
-Started creating textures and converting them into mdls for use in game
Object examples:
The Chroma Cycle
Puzzle switch green
Puzzle switch yellow
The pushing block
Parent's fountain
The pillar base and the pillar column

12/4/05
-Made a rock stalactite model and texture for the pit room
-Tuned up the menu screen

12/7/05
- Worked on getting objects and textures into mdl form
- Started to place objects into level

12/9/05
- Colored some sketches for the presentation
- Continued texturing mdls and interior decorating the game level

8.7.4 :: Alex Tanzio
The journal that never was... but now is...

Week 1
Worked on game concept.

Week 2
More game concept work as well as concept art of WooWoo Nut and Gentry Chroma relic hand blaster. No one liked the idea of a hand blaster *tear*

Week 3
Decided on using GameStudio and started messing around with template code. Tried working with MilkShape and other importing programs to get models into GS. None of them seemed to work.

Week 4
Templates don’t seem to be the way to go with GS. They make no sense. Custom coding is needed. Gathered different tutorials to help with custom code. Maya to GS plug-in seems to be the way to get things into GS with animations.

Week 5
Continued to work with GS and making a playable demo.

Week 6
Camera was custom coded as well as player movement. Some enemy interactions begun but hey don’t work completely yet.

Week 7
Enemy interactions worked out. Still using the GS models but Fistucko has been imported. Game engine can handle a large amount of them so the way we are modeling them seems to be ok. Looked into level creation add-on for 3D Studio Max.

Week 8
Firing and four chroma shots implemented. Difference of them can’t be seen but when the right one is fired it will destroy the
enemy. Multiple enemies working at once and player health working.

**Week 9**
Not a ton done due to Thanksgiving break. Additional coding worked on and current code commented.

**Week 10**
Just worked on more coding various things implemented. Created design documentation for the existing code.

**Week 11**
With all assets available game came together. Matt and I tore into all the coding that needed to be done and got nearly everything done from animations to level interactions and everything else.

**Week 12**
Commented all code that needed to be commented and made organizational changes to area of code. Finishing up and last touches ups that need to be made for final presentation like pause and such so presenting is easier.

**8.7.5 :: Pat Xin**
**Temporal dated late 9.05**
- Art began in game process Alpha Gentry synthesized.
- All enemy concepts complete.

**10.3-9.05**
- Zoop as inspiration modeled basic concept. Worked around the concept initial art and all.

**10.10-16.05**

**10.17-23.05**
- Meeting for idea to restructure game play
  Structure of Rockimichili formed and modeling started

**10.24-25.05**

- Continue until finished with masks and Rockimichili, argument with Smith about shooting fireballs out of nose.

**10.30.05**
- Level concept finished overhead only.

**11.2.05**
- Final concept for all Enemies done.

**11.9.05**
- Side concept of initial room done.

**11.12.05**
- Concept for Rockimichili and ideas for other rooms started.

**11.16.05**
- Revamped Rockimichili art concept.

**11.20.05**
- Finalized enemy AI with concept of getting the enemy to follow and rotate at gentry.

**11.26.05**
- Concept of Mokai Box for interacting with enemy AI finished with Alex.

**11.17.05**
- Texture of floor, walls, columns done.

**11.29.05**
- Worked on roof and door textures.

**12.3.05**
- Worked on mechanics of a circular camera.
- Finished Door texture, Roof texture...

**12.7.05**
- WooWoo nut concept done.
- Worked on New game art.
- Worked on brochure sheets.

**12.9.05**
- Game Art done finished uncolored
- Character sheet art 40% done, uncolored.

**Current:**
- Working on coloring character sheet (50% done and colored) and Rocki.
- Will make tile-able moss overlay texture.
- Working on symbols, so far only room clear and slow wheel done.
  - (possible) Concept for a later room.

End Transmission.

**8.7.6 :: Josh Landis**

**Week 1**
Gathered at the brainstorming session for our game ideas.

**Week 2**
Assisted with refocusing game idea. Began working on 3D work for
design of rooms. Used concept art to build a functioning color wheel
to help us get the wheels timing down.

**Week 3**
Began work on our dog enemy, known as the Fistucko, using Pat’s
designs for inspiration. Used Maya 6.0 to box model the creature,
so he could be rigged and animated later by Matt Smith. Completed
modeling of Fistucko for Matt, and handed it off to him with its
bone structure. Attempted to use MotionBuilder to help Smith out,
but was unsuccessful with naming conventions and rigging.

**Week 4**
Began work on Gentry, our main character, again using Pat’s
drawings. Completed Gentry. Now had the experience to build
Gentry’s bones and bring him into MotionBuilder. Used motion
capture data to get smoother movements to illustrate his cartoony
anatomy.

**Week 5**
Because of crits involving our dog character, I realized he needed to
be reworked to be a bit less Lassie, and a bit more Cujo. I made the
changes to geometry, but found that the animations would need to
be redone. Also made some changes to Gentry, but had no issues
getting the new version onto Matt’s animations.

**Week 6**
Brought the creature into MotionBuilder and made his rig so he
could be more easily animated. First noticed an issue with the
skeleton moving incorrectly.

**Week 7**
After cursing the day MotionBuilder 6 was released, I spent a great
deal of time attempting to fix an issue with the Fistucko’s rig.
Something about the rig was causing his front legs to bend
incorrectly, thus creating a strange problem. Also began looking
into UV texture editing at this time and began work on laying out
the Fistucko’s UVs.

**Week 8**
On a whim, I asked Ari to try the Fistucko out in the newest version of MotionBuilder, version 7. Lo and behold, the rig now worked correctly. I fixed the rig, as well as built several items for inside our level.

**Week 9**
I began work on laying out and texturing our characters, which during our other classes involving modeling, was vaguely mentioned. Also built more items for inside the level, including vases and statues for each of the creatures to be placed.

**Week 10**
I began work on laying out and texturing our characters, which during our other classes involving modeling, was vaguely mentioned. I then bumped into the wonderful phenomenon known as “swimming UVs”. I was able to rebind the skeletons after the UVs were set, thus fixing the issue.