WORCESTER POLYTECHNIC INSTITUTE

Interactive Media and Game Development

Final Report

for the

Design of Blood Tide: A Multiplayer Online Action RTS

by

Joseph Alea, Garret Doe, Sarah Gilkey, Chris Williams, Frank Williams

Major Qualifying Project
October 13, 2009

DESIGN ABSTRACT

Blood Tide is a multiplayer, online, action, real-time strategy game. It allows the player to experience both the strategic management of an army and the fast-paced, high-action, combat on the battlefield alongside other troops. *Blood Tide* will pit two players against each other in an underwater combat scenario. Games of this unique genre, a combination of Action and Real-Time Strategy (RTS), are uncommon and upcoming in the game development industry. Team Blackfire is creating *Blood Tide* based on our own vision and ideas to pioneer this genre of game.

This document fully describes every gameplay feature, the technical components required for the game to function, and finally the artistic vision and theme. Chapter 2 contains the game overview, as well as a one sentence descriptor, to give the player a good understanding of the game story and the objectives of *Blood Tide*. Chapter 3 describes the entities in the game, like Bases, Buildings, and Unit names and abilities, and how the player will create them considering the different gameplay modes. Chapter 4 fully explains how the Action mode, RTS mode, and the transition between them will work. Chapter 5 delves heavily into the artistic theme for the game, providing both pictures that inspired ideas for features of the game, and concept art as a precursor to the models and textures. Chapter 6 provides the logistics behind why specific game engine and tools were used. Every gameplay mechanic is described technically from the programming point-of-view, from explaining unit artificial intelligence to the tools built for customizable and modification purposes. Both Chapters 5 and 6 display our due dates in the form of a timeline for when we want specific features of the game to be finished. Such timelines will keep our team constantly on task and organized.

Team Blackfire is excited to have critically developed *Blood Tide* in such detail, ready to flesh out in a 3D world. The entire game is described in this document; gameplay ideas may be tweaked slightly during development, but our team is confident in this document to provide a firm, complete blueprint for the building process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

1.1 ONE SENTENCE

Whether you want to command an entire army or fight as an elite field-commander alongside your compatriots, *Blood Tide* unites strategy and action players against the opposing team in an underwater battle for supremacy.

1.2 ONE PARAGRAPH DESCRIPTION

War on Earth has rendered its surface uninhabitable for the human race. In order to preserve themselves, humans have moved to the depths of the ocean and established new colonies. However, the humans' fight for survival is threatening the delicate ecosystem which descendants from the lost city of Atlantis have survived within for millennia. Neither side is willing to compromise over the limited underwater resources, so the two factions are gearing up for battle – a battle that will determine which race will thrive and which will become a memory. *Blood Tide* is a multiplayer-online-action-real-time-strategy-game that allows players to take the role of a tactical Commander or elite field-commander for two different factions.

2 GAMEPLAY OVERVIEW

Blood Tide is a multiplayer game which pits two players against each other. Each player will take control of either the Humans or Piscivian faction. Gameplay takes place in two modes: command mode and field mode. In command mode, the player can control units and construct buildings, like an RTS game. In field mode the player can take the role of the field-commander for a faction. This mode plays like an action game with a third-person chase camera. The player can switch between the two modes to better utilize the functions of the field-commander and commander.

At the start of a game session, each faction has three bases. These bases act as spawning points for different types of units that can be further specialized through the building of farms, training grounds, and guard towers. When a faction loses all three of its bases, that player loses the game.

Winning a game is determined by how effectively the player can manage units, the strategy employed when upgrading bases, and how well the player controls the field-commander in skirmishes.

3 GAMEPLAY DETAIL

This section discusses the game detail mechanics.

3.1 BASES

At the start of a game session, each faction begins with three bases. Each base spawns a certain type of unit nearby. A base has a set number of hit points, and once destroyed, no longer spawns units, thus effectively removing that type of unit from that faction's future reinforcements. It cannot be rebuilt, so the player must protect each base to continue providing the faction with a variety of units.

Around each base are a set number of terrain plots, upon which the player can construct buildings. This is done by selecting a base, then the plot, and finally the building type.

3.2 BUILDING TYPES

A building can either be a defensive structure or one that provides a passive bonus for the units that spawn from that base. Buildings take no resources to construct—only time. Only one building can be in constructed at a time. Once the building is complete, the player is notified and is able to begin constructing a new building. If a building is destroyed, the bonus that it once granted is removed and the plot becomes available for building again. However, if the base that all the building plots belong to is destroyed, then all the plots are off limits for construction for the remainder of the game session. The types of buildings the player can construct on plots of land include:

- Guard Tower—a defensive structure that automatically attacks enemy units in range.
- Farm—a structure that causes units to spawn at the base in larger numbers.
- Training Facility—a structure that boosts the strength of existing and future units.

3.3 UNITS

The following tables describe the three units for each faction. All unit types have three abilities: melee attack, ranged attack and support ability. The stars shown next to each field in the stats category represent the value of that particular statistic—more stars mean a higher value.

TABLE 1: PISCIVIAN FACTION UNIT ABILITIES AND STATS CHART

Piscivian Faction			
Unit	**Merman**	**Manta Ray**	**Turtle**
Abilities	**Trident** - basic melee attack with trident *(Melee)* **Laser Trident** - a magic beam of light fires from trident *(Range)* **Serenade** - a tribal song that gives healing bonuses to all in range *(Special)*	**Tail Whip** - the manta will spin around whipping surrounding units with its tail *(Melee)* **Stealth** - hides from enemy site for a limited time, can bypass guard towers *(Special)* **Whirlpool** - creates an underwater tornado *(Range)*	**Wave Cannon** - fires a wave *(Range)* **Shell Barrier** - nullifies all damage done to the turtle *(Special)* **Bite** - a strong bite attack that pierces through enemy defense *(Melee)*
Stats	**Hit Points**: ★★★ **Strength**: ★★★ **Armor**: ★★★ **Movement Speed**: ★★★ **Attack Speed**: ★★★	**Hit Points**: ★★ **Strength**: ★★★★ **Armor**: ★ **Movement Speed**: ★★★★★ **Attack Speed**: ★★★	**Hit Points**: ★★★★ **Strength**: ★★ **Armor**: ★★★★★ **Movement Speed**: ★★ **Attack Speed**: ★★

TABLE 2: HUMAN FACTION UNIT ABILITIES AND STATS CHART

Human Faction			
Unit	**Scuba Troop**	**Submarine**	**Bomb Ship**
Abilities	**Hand Blade**- basic melee attack with spinning propeller blade*(Melee)* **Harpoon Gun** - fires a harpoon *(Range)* **Speed Boost** - enables scuba troop to quickly propel through water *(Special)*	**Torpedo** - fires a torpedo *(Range)* **Radar** - increases scouting abilities by expanding range of site *(Special)* **Excavation Drill** - a drill made for tearing down buildings *(Melee)*	**Cannon** - fires a an explosive cannon ball *(Range)* **Bomb** - plant shrapnel that will explode after a few seconds *(Melee)* **Self-Destruct** - detonates the ship and destroys everything in the explosion *(Special)*
Stats	**Hit Points**: ★★★ **Strength**: ★★★★ **Armor**: ★★ **Movement Speed**: ★★★ **Attack Speed:** ★★★	**Hit Points**: ★★ **Strength**: ★★★ **Armor**: ★★★★ **Movement Speed**: ★★★★ **Attack Speed**: ★★	**Hit Points**: ★★★★★ **Strength**: ★★★★★ **Armor**: ★★★ **Movement Speed**: ★ **Attack Speed**: ★

3.4 FIELD-COMMANDER

At the start of the game session, the player has the ability to choose which type of field-commander to play as. Each base will spawn a different type of field-commander that coincides with the regular units that spawn from that base. For example, the turtle base will spawn a turtle field-commander. If a player wishes to change to a different field-commander type, the player must move the field-commander back to the faction base it came from and choose a different type.

3.5 UNITS VS. FIELD-COMMANDER

All units and the field-commander will start with one attack. For instance, both a unit and field-commander bomb ship will start off with the bomb ability; however, only the field-commander is capable of unlocking the next two abilities for the unit type. These abilities are unlocked for the field-commander upon promotion (*Field-commander Ranks*). The field-commander is stronger than its normal unit counterpart. When a training ground is built, both the field-commander and the normal units of that type will be upgraded.

4 GAMEPLAY MODES

At the start of a game session, the player plays the game in command mode. The goal of this mode is for the player to strategically control the faction's units and to specialize bases and units through the construction of buildings. In this mode, the player can select the type of field-commander that spawns. In command mode, the player cannot control the field-commander. The player must switch to field mode in order to control it. The goal of field mode is to utilize the strengths and abilities of the chosen field-commander and help the normal units in skirmishes. The player can alternate between gameplay modes.

4.1 COMMAND MODE

In this mode, the player has the ability to select units, issue orders to those units and construct buildings. The mode uses an overhead camera to give the player a birds-eye view of the map. It is played like an RTS game, utilizing the mouse to pan the camera around the map, select game entities, and issue orders. The player will be able to customize the control scheme by using the menu system that comes with the C4 engine.

4.1.1 USER INTERFACE

The command-mode UI (**Error! Reference source not found.**) consists of four elements:

- Chat box: used for chatting with the other player.
- Group-information panel: contains information about a group of units that was selected.
- Unit-information panel: contains more detailed information about a single unit: the unit's health and status. The unit information panel will be updated when a single unit is selected, or it will provide information for the first unit to show up in the group information panel
- Orders panel: contains buttons for all of the orders (move, attack and stop) that units can carry out.

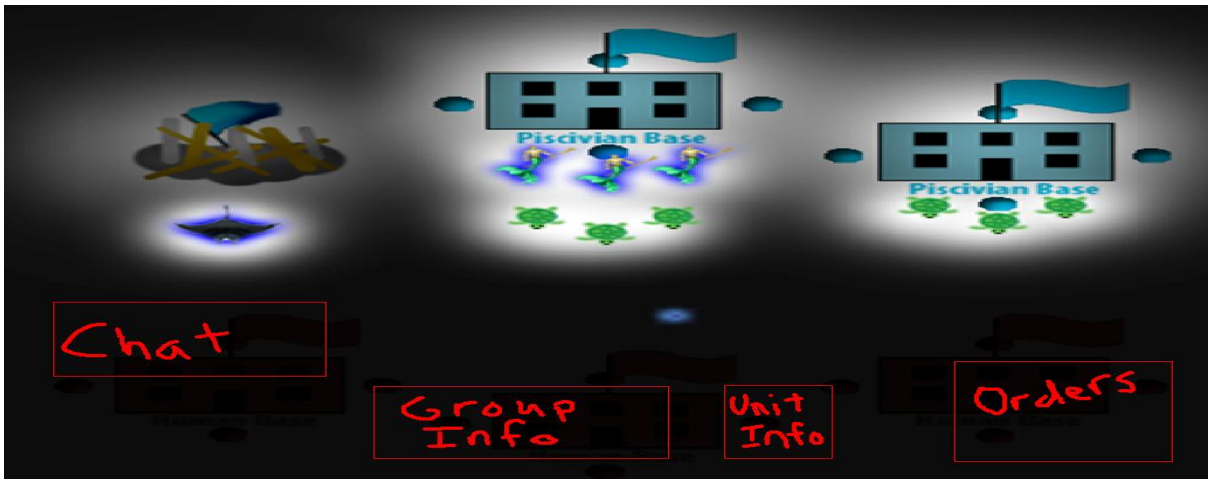


FIGURE 1: COMMAND MODE UI

4.1.2 UNIT SELECTION

All unit selection will be done using the left mouse button. Left-clicking on a unit will select that unit. Clicking on that unit again will select all units of that type in some range. Clicking on that unit again or on any other unit will select that unit and clear the group information panel. Also, the player can hold the left-button down, which will create a select-box that the player can drag around. When the left-button is released, all units in the select-box will be selected.

4.1.3 UNIT ORDERS

When units are selected, the player may issue orders to them via the orders panel or the keys the orders are mapped to. The player can issue three orders:

- Move: moves all selected units to a selected destination.
- Attack to: works the same as the move order, except that when enemy units are spotted, they will be attacked and the units will move to the selected destination after the fight.
- Stop: halts all orders the units were following.

4.1.4 BUILDING UI

When a player selects a plot of land, an interface will pop up that displays all buildings that can be constructed on the selected plot. If a building cannot be built, it will be grayed out.

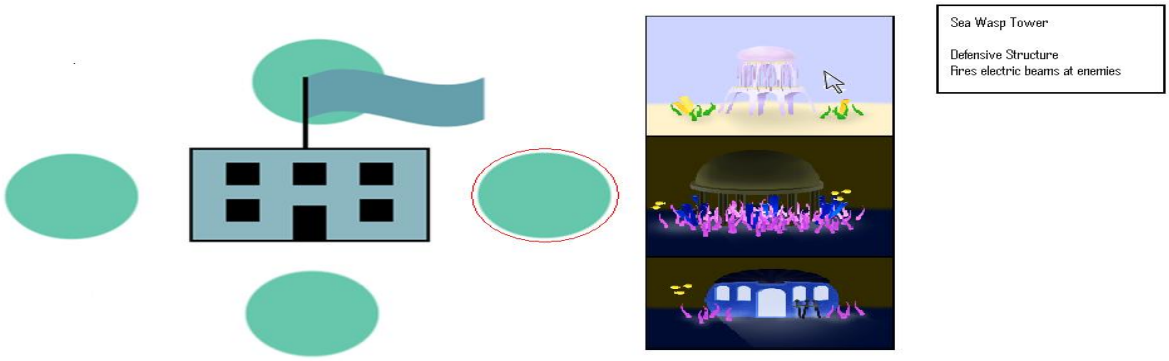


FIGURE 2: BUILDING UI

4.2 FIELD MODE

In this mode, the player takes control of the faction's field-commander. It is played like an action game with a third-person chase camera. Play will default to using the WASD control scheme with mouse look. The player will be able to customize the control scheme by using the menu system that comes with the C4 engine.

4.2.1 FIELD-COMMANDER RANKS

The field-commander will start at rank one. In order to get promoted, the field-commander must gain a certain number of combat points; these combat points are gained by killing enemy units, bases, buildings, or the opposing field-commander.

There will be two additional ranks for the field-commander to gain. When the field-commander is promoted, he/she will become stronger and gain a new ability or attack. When the field-commander dies, the next field-commander chosen will spawn at rank one without any active upgrades.

4.2.2 USER INTERFACE

The field mode UI (**Error! Reference source not found.**) consists of four elements:

- Field-commander information: consists of the field-commander's health, combat points, and rank.
- Alert panel: provides information to the player; it will display a camera viewing the relevant object. For instance, if a building just finished construction, the alert panel will display a view of the completed building.
- Chat box: used for chatting with the other player.
- Abilities panel: provides all abilities for the given field-commander type; abilities that have not been unlocked will be grayed out.

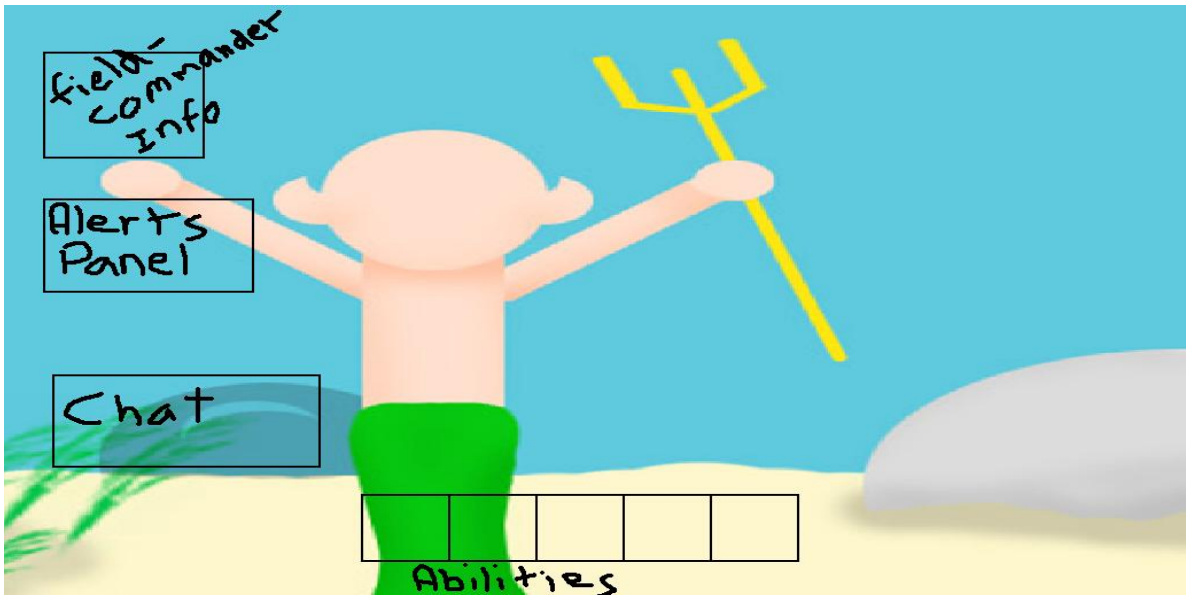


FIGURE 3: FIELD MODE UI

5 ART DESIGN

The art in *Blood Tide* will complement the aquatic theme with a combination of realistic and fantastic styles while keeping a serious mood during the game. The Piscivian and Human factions will each have their own art styles.

The art style for the Piscivian faction will blend Doric Greek architecture with a Hawaiian white-sand beach theme. We were inspired by the feel and familiarity of ancient Doric Greek architecture, but we wanted to blend the Hawaiian white-sand beach theme to create a more unique art style. Additionally, we wanted the Piscivians art style to harmonize with *Blood Tide's* story; therefore, the Piscivian art style needed to represent a lush, natural, and delicate ecosystem which the Humans are threatening.

The art style for the Human faction will blend World War II industrial style with the underwater architecture and design of the Hydropolis Hotel[1] – a planned underwater hotel, scheduled to be opened by the end of 2009, which will be constructed off Jumeira Beach in Dubai (Figure 4). We were inspired by the familiarity of World War II weaponry and military bases seen in other war games. However, we wanted to blend a futuristic architecture that would represent a humanized underwater city.

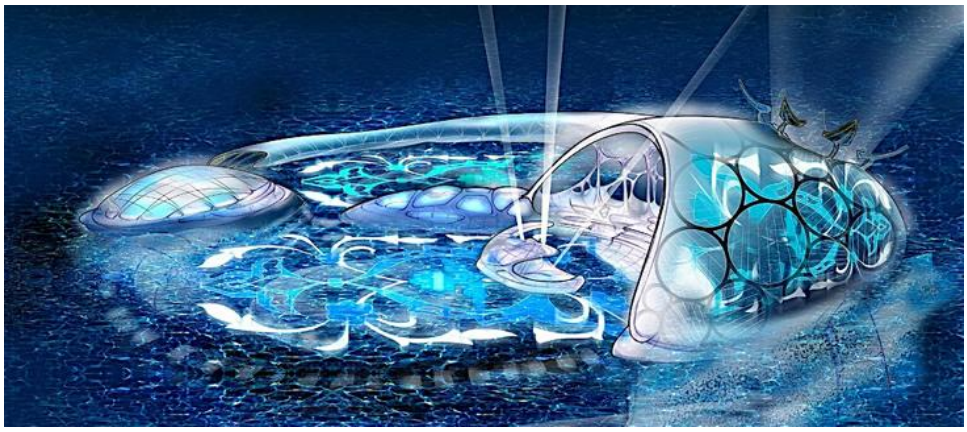


FIGURE 4: HYDROPOLIS HOTEL

This section will cover all the art assets that are expected to be in *Blood Tide* and how each asset corresponds to the overarching theme for each faction. There are several different types of art assets that we will include in our game: 2D art, 3D models, and audio.

[1] Refer to http://www.hydropolis.com/ (October 12, 2009)

5.1 2D ART

The 2D art in our game will be comprised of menus. We will be using *Adobe Photoshop* to create the 2D art for our game.

5.1.1 MAIN MENU

The main menu for our game will consist of the game title, our team name, and the option to begin a multiplayer battle using either the Piscivian faction or Human faction. The main menu will follow the red and blue color scheme in *Blood Tide*. The options on the main menu will consist of Host Game, Join Game, Settings, and Quit (Figure 5).

FIGURE 5: BLOOD TIDE MAIN MENU

5.2 3D ART

The 3D art in our game will comprise of units, buildings, and environment. We will be using *Autodesk Maya* to create all the 3D art.

5.2.1 PLAYABLE UNITS

There are six different units in *Blood Tide* with three for each faction. Each unit will specialize in a specific attack type (melee or ranged) and will have different strengths and weaknesses (Table 3).

The units of the Piscivian faction will be more agile while the units of the Human faction will be stronger.

TABLE 3: UNIT STRENGTH AND WEAKNESS CHART

Unit	Faction	Health	Strength	Armor	Movement Speed	Rate of Attack
Merman	Piscivian	★★★	★★★	★★★	★★★	★★★
Manta Ray	Piscivian	★★	★★★★	★	★★★★★	★★★
Turtle	Piscivian	★★★★★	★★	★★★★★	★★	★★
Scuba Troop	Human	★★★	★★★★	★★	★★★	★★★
Submarine	Human	★★	★★★	★★★★	★★★★	★★
Bomb Ship	Human	★★★★★	★★★★★	★★★	★	★

When the player chooses a field-commander, he/she will be able to use a base ability for the respective unit. In addition, the player will be able to unlock two additional abilities at the highest promotional rank.

5.2.1.1 PISCIVIAN UNITS

There are three different units to command on the Piscivian faction, and each unit will have its own strengths and weaknesses. When starting a battle session, the player will be prompted to select a type of Piscivian unit as the field-commander.

MERMAN

Mermen are the most balanced unit in the game and specialize in close combat using a trident as a weapon. The merman unit will resemble mermen portrayed in Disney's *The Little Mermaid* - an attractive, bearded man with a fish tail (Figure 6). The fish tail will be shaped and colored a dark blue. This will resemble the Picasso triggerfish, which is the state fish of Hawaii. The human half of the merman will have short brown hair and a short full beard. He will be built like a swimmer and have broad shoulders and thin hips. The transition from human to fish will be done abruptly like it is done in *The Little Mermaid*.

FIGURE 6: MERMAN

MANTA RAY

A manta ray is a stealthy unit with strong attack and low defense that specializes in close combat melee using barbed tail. Manta rays live in the coral reefs off the coasts the Hawaiian Islands. The manta ray unit will have a shiny black top and a cream underside (Figure 7).

FIGURE 7: MANTA RAY

TURTLE

Turtles are highly defensive units with low speed and strength and specialize in ranged combat using a wave cannon. The turtle unit will look similar to the common green turtle found off the coast of Hawaii, but will have a large cannon tied to its shell (Figure 8). The cannon will look like a conch shell, which will blow water waves.

FIGURE 8: TURTLE

5.2.1.2 HUMAN UNITS

There are three different units to command on the Human faction, and each unit will have its own strengths and weaknesses. When starting a battle session, you will be prompted to select a type of Human unit as your field-commander.

SCUBA TROOP

Scuba troops are balanced units that specialize in melee attacks using two hand blades. The scuba troops will look similar to traditional scuba divers (Figure 9) but will have a propeller in each hand to increase movement through water. They will have red and black wetsuits.

FIGURE 9: SCUBA DIVER

SUBMARINE

Submarines are fast and defensive units that specialize in ranged attacks with torpedoes. Submarines will have a poor rate of fire for their torpedo launcher. Submarines will be small personal vessels with a body that resembles WWII submarines (Figure 10). The submarine will have a clear metal coating, similar to the glass architecture seen for the Hydropolis Hotel, with titanium binding the submarine together.

FIGURE 10: WWII SUBMARINE

BOMB SHIP

Bomb ships are powerful but extremely slow units that specialize in ranged attacks with bombs. Bomb ships will resemble aircraft carriers during World War II. Bomb ships will borrow the flat top of air craft carries and the immense size. The bomb ship will have protruding titanium spikes at the front of the ship which resemble the crown-like structure on top of the Hydropolis Hotel. Additionally, bomb ships will have several shiny fins similar to architecture seen on the Hydropolis Hotel, on port, starboard, and aft (Figure 11).

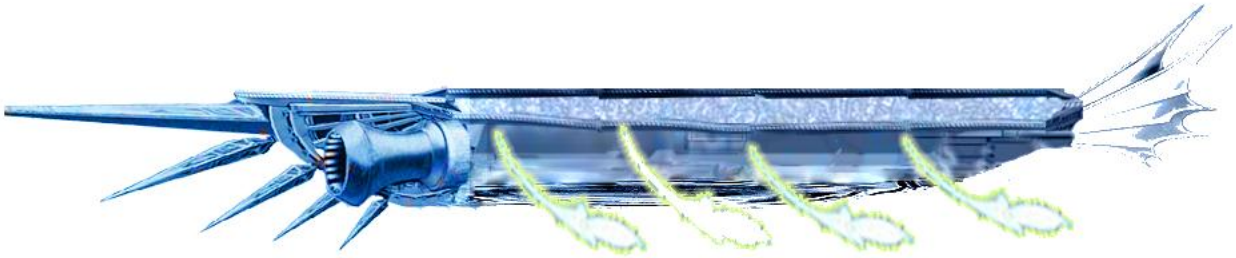


FIGURE 11: BOMB SHIP

5.2.2 BUILDINGS

Each of the three units for the Piscivian and Human factions will have its own unique base in which units will spawn. Additionally, the player may build defense towers, farms, and training facilities to improve defense at a base, increase the number of units spawned, and increase the quality of units, respectively.

5.2.2.1 PISCIVIAN BUILDINGS

The art theme of the Piscivian faction's buildings will be influenced by ancient Doric Greek architecture blended with Hawaiian beach them. This will incorporate undersea materials such as seashells, starfish, and sand into the Doric structures.

MERMAN BASE

The merman base will be a tall tower that resembles a sandcastle made of white sand. In this sandcastle will be imprints of shells and starfish. At the top of the base will be a triangular roof noticeable in Doric Greek architecture. Additionally, the merman base will be surrounded by several Doric Greek pillars (Figure 12).

FIGURE 12: MERMAN BASE

MANTA RAY BASE

The manta ray base will be a small coral reef with various types of plant life and organisms living on it (Figure 13).

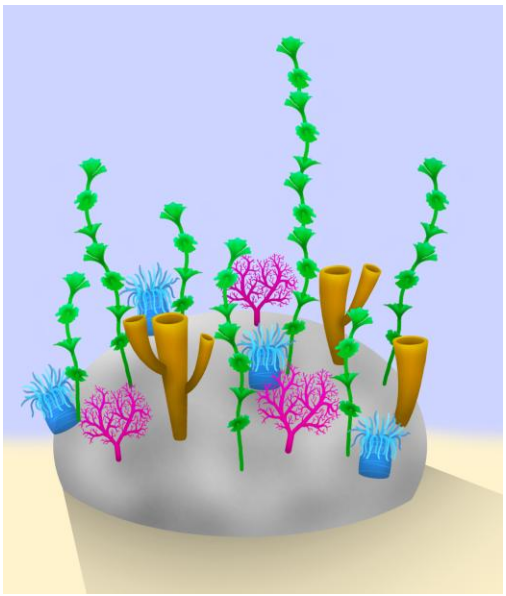


FIGURE 13: MANTA RAY BASE

TURTLE BASE

The turtle base will resemble an open underwater garden containing many clam shells. The garden will be surrounded by Doric pillars which will be tangled in kelp and other sea grasses.

SEA WASP TOWER

The Piscivian defense tower will resemble a tall Doric pillar. On the top of the pillar will be a giant jellyfish which will fire electric beams at enemy forces. We chose to name the tower after the box jellyfish (nicknamed sea wasp), which is considered a deadly jellyfish found in parts of Hawaii (Figure 14).

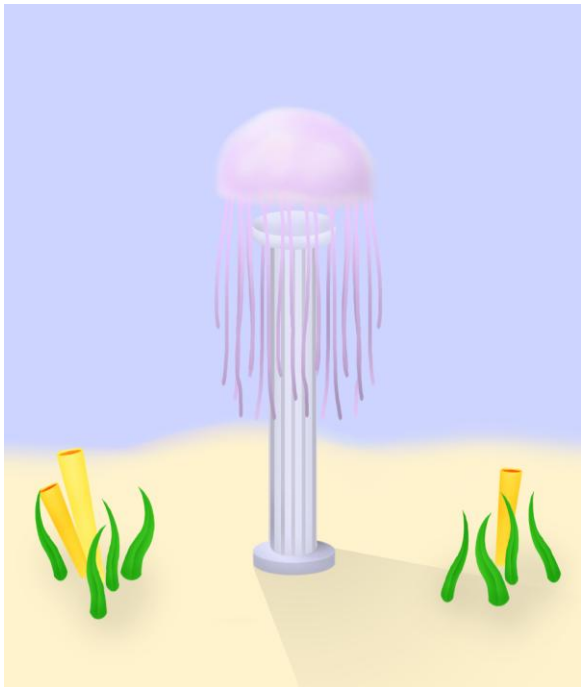


FIGURE 14: SEA WASP TOWER

KELP FARM

The Piscivian Kelp Farm will be a sun starfish that is held by several Doric pillars. Underneath the structure will be several aquatic plants and sea grasses that are found in the Hawaiian region (Figure 15).

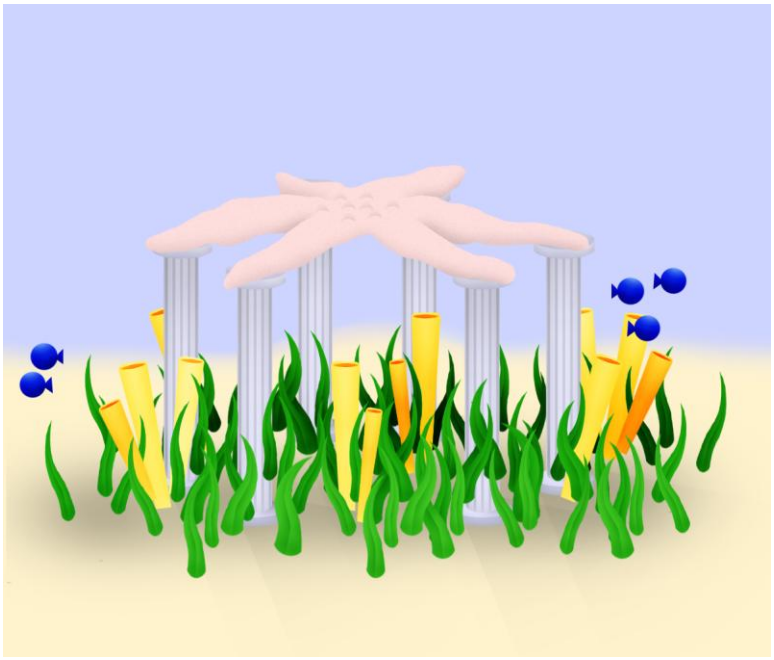


FIGURE 15: KELP FARM

BATTLE ARMORY

The Piscivian training facility will resemble a Doric temple (Figure 16) and will have a giant shell on the top of the building.

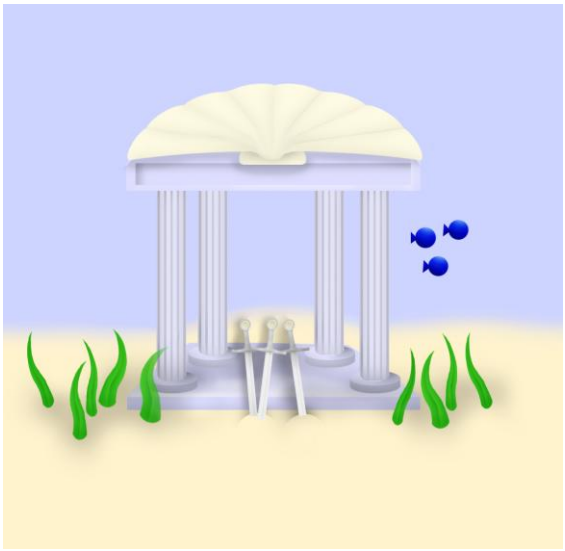


FIGURE 16: BATTLE ARMORY

5.2.2.2 HUMAN BUILDINGS

The art theme of the Human faction's buildings will be influenced by World War II weaponry and familiar architecture blended with the futuristic, aquatic city feel of the Hydropolis Hotel.

SCUBA TROOP BASE

The scuba troop base will be an underwater station that resembles the Hydropolis Hotel (Figure 4).

SUBMARINE BASE

The submarine base will resemble a World War II sub pen having the long tunnels in which submarines are docked (Figure 17). The submarine base will be a circular structure constructed out of shiny metals and glass.

FIGURE 17: WWII SUB PEN

BOMB SHIP BASE

The bomb ship base will resemble a long, glass, titanium building. The building will have docking stations on which the bomb ships will spawn. The docking stations will look similar to harbors used in World War II, but with the futuristic architecture similar to the Hydropolis Hotel.

TURRET

The Human defense tower will look similar to a World War II turret. The turret will have six barrels designed for small torpedoes (Figure 18).

FIGURE 18: WWII TURRET

FARM

The Human farm will resemble the hemispheric glass dome structure on the Hydropolis Hotel. The hotel has placed greenery and trees inside and the farm will contain a fielded green area and giant tree.

TRAINING BUNKER

The Human training facility will resemble a battle bunker used during World War II (Figure 19). However, instead of the cement exterior, the bunker will be made from a clear metal coating and titanium architecture used for the Hydropolis Hotel (Figure 4)

FIGURE 19: WWII BATTLE BUNKER

5.2.3 ENVIRONMENT

The battle between the Humans and the Piscivians occurs on the ocean floor of Earth. The level will contain aquatic scenery including coral reefs, bubbles, kelp, underwater ruins, and schools of fish. The scenic environment will be included to further immerse the player in the underwater setting. In addition, the environment will be used to create natural boundaries around the battleground. We will use ocean drop-offs, geysers, cliffs, and dangerous undersea life such as jellyfish.

5.3 AUDIO DESIGN

We will have two varieties of audio in our game – music and sound effects. We will write custom game music using Finale Notepad, which is regarded as the industry standard, and we will collect sound effects using a recorder and adjust them in Audacity.

5.3.1 GAME MUSIC

The game will have three different musical tracks.

The first song will be used in the menu system of *Blood Tide*, before a player enters the game. This song will be melodic. This piece will use piano, harp, and flute to welcome the player.

The second song will be played while the player is in command mode. It will resemble *The Battle* by Harry Gregson-Williams used in the movie *Narnia: The Lion the Witch and the Wardrobe*. This music piece consists of mostly strings and horns.

The third song will be played while the player is in field mode. It will be a faster, remixed version of the command mode song and incorporate choir, timpani, and drums.

5.3.2 SOUND EFFECTS

This section details all of the sound effects that will be included in *Blood Tide*.

5.3.2.1 GENERAL SOUNDS

The general sounds will be common to all units, buildings, and factions. These sound effects will be heard while the player is in command mode or when the player is close to the source of a sound effect during field mode.

UNIT SPAWN

The unit spawn sound will be used when a wave of units have spawned. This sound will repeat for every new wave of units. This will sound similar to a bell chime.

FIELD-COMMANDER SPAWN

The field-commander spawn sound will be used when the field-commander has re-spawned after death. This will sound like a choir arpeggio.

ORGANIC UNIT DEATH

The organic unit death sound will be used when any non-mechanical NPC unit dies. This sound effect will be used for the Merman, Manta Ray, Turtle, and Scuba Troop units. A burst of bubbles sound is played when an organic unit dies.

MECHANICAL UNIT DEATH

The mechanical unit death sound will be used when any non-organic NPC unit dies. This sound effect will be used for the Submarine and Bomb Ship units. This will sound like a small underwater explosion.

5.3.2.2 BUILDINGS SOUNDS

Building sounds will be heard while the player is in command mode or when the player is close to the source of a sound effect during field mode.

BUILDING CREATED

The building created sound will be used to alert the player whenever a building has finished construction. This will sound like a low frequency chime.

BUILDING DESTROYED

The building destroyed sound will be used whenever a building or base is destroyed. It will sound like a small explosion.

SEA WASP ATTACK TOWER

The Sea Wasp Tower will make a zap sound when attacking. There will be a crackling effect after the initial shot, similar to a bug zapper.

TURRET ATTACK

The Turret will sound like cannon shooting torpedoes. It will sound like a small explosion combined with the rushing of water.

5.3.2.3 UNIT SOUNDS

Each unit type will have three different sound effects – movement sound, attack sound, and accept order sound. The movement and attack sounds can be faintly heard when in command mode; however, while in field mode the sounds are heard when in range of the source. The accept order sound effects are only heard while in command mode. See Table 4 for a complete list of sound effects for the Piscivian faction units and see Table 5 for sound effects for the Human faction units.

TABLE 4: SOUND EFFECTS FOR PISCIVIAN UNITS

Piscivian Faction			
Units	**Move SFX**	**Attack SFX**	**Accept Order SFX**
Merman	This sound effect will be used whenever the merman is swimming around the map. This will sound like a fish swimming through water, and have a water rush sound.	The attack sound effect will be used when the merman unit is attacking an enemy unit or building. This will sound like a swish followed by a rushing water sound.	The merman will signal that he has received orders by playing the accepting orders sound effect. It will play a small song-like choir chant.
Manta Ray	This sound effect will be used whenever the manta ray is moving around the map. A water rushing sound effect will be played when the manta ray flaps its wing-like fins.	The attack sound effect will be used when the manta ray unit is attacking an enemy unit or building. This will sound like a swish followed by a whip cracking sound effect.	The manta ray will signal that it has received orders by playing the accepting orders sound effect. Bubbles followed by rushing water will designate that the manta ray has accepted an order.
Turtle	This sound effect will be used whenever the turtle is swimming around the map. A sound of water rushing will be played when the turtle flaps its fins.	The attack sound effect will be used when the turtle unit is attacking an enemy unit or building. An inhaling noise will play to charge the cannon and will follow with a swish with an echo.	The turtle will signal that it has received orders by playing the accepting orders sound effect. Bubbles followed by rushing water will designate that the turtle has accepted an order.

TABLE 5: SOUND EFFECTS FOR HUMAN UNITS

Human Faction Sounds			
Units	**Move SFX**	**Attack SFX**	**Accept Order SFX**
Scuba Troop	This sound effect will be used whenever the scuba troop is moving around the map. A faint buzzing sound of propellers will be played as the scuba troop moves around the map. Additionally, the player will hear bubbles from the air tank when using a scuba troop as a field-commander.	The attack sound effect will be used when the scuba troop unit is attacking an enemy unit or building. The sound of a razor blade cutting wood will play when the scuba troop attacks. A sound of rushing water will follow.	The scuba troop will signal that he has received orders by playing the accepting orders sound effect. A heavy breathing noise, similar to the sound Darth Vader makes, will sound when the scuba troop accepts an order. This will follow by bubbles from the air tank.
Submarine	This sound effect will be used whenever the submarine pilots around the map. A faint buzzing sound of propellers will be played as the submarine moves around the map.	There will be no sound for when the torpedo is fired, because a torpedo breaks the speed of sound. However, there will be a small explosion sound that is played after a target is hit.	The submarine will signal that it has received orders by playing the accepting orders sound effect. A submarine sonar sound will play when the submarine accepts an order.
Bomb Ship	This sound effect will be used whenever the bomb ship is maneuvering around the map. A swishing noise and water rushing noise will sound when the bomb ship moves its metal fins to move through the water.	The attack sound effect will be used when the bomb ship unit is attacking an enemy unit or building. A cannon sound will fire when the bomb ship attacks. An explosion will sound when the bomb ship hits the target.	The bomb ship will signal that it has received orders by playing the accepting orders sound effect. A human will communicate back saying “For Freedom” or “Affirmative.”

6 TECHNICAL DESIGN

This section outlines the technical aspects of the development of *Blood Tide*. Specifically, this section includes the tools that the team will be using to create the game, the target technical performance of the game, the planned components that will need to be built for the game, and the projected milestones for the development of the game.

6.1 ENGINE CHOICE RATIONALE

This project will be based on the C4 Engine from Terathon Software LLC.[2] The project will use engine-level code, such as graphics rendering, user input, data structures, and make use of portions of non-gameplay related code. The non-gameplay related code, which can be found in the tutorial game that comes packaged with the C4 Engine distribution, includes high-level networking capabilities, basic character controllers, cameras, basic user interface mechanisms, and projectiles.

One of the two major reasons why the project will use the C4 Engine is that it supports technical features that are needed for this project, such as:

- A well-designed, object-oriented engine that encourages the team to use good design practices.
- Access to the entire code-base, including engine-level code, so the team can modify or research any part of the engine.
- Built-in networking capabilities, allowing the team to create synchronization mechanisms without being concerned for underlying networking functionality.
- Support for importing art assets in an industry standard COLLADA[3] file, allowing the artists to use a variety of content creation applications.
- Support for a variety of lighting techniques and special effects, creating particle effects based on existing systems.
- Customizable world editor and art asset viewing utilities, supporting the simplistic creation of levels.
- A structured approach to creating user interface elements.

The other major reason for using the C4 Engine is that the team is familiar with the engine; every member on the team has used the engine for programming or creating art assets. There are many reasons why familiarity with the engine will help this project. Most importantly, the team will not have to spend time learning how to use the engine. This allows the team to allocate more time to

[2] Refer to http://www.terathon.com/ (September 21, 2009)
[3] Refer to https://collada.org/ (September 21, 2009)

create art assets and program more complex gameplay mechanisms, such as advanced artificial intelligence algorithms or advanced optimization techniques. Also, the team is familiar with ways to get help, such as the official C4 Engine forums, of which the developers of the engine regularly participate.[4]

6.2 PERFORMANCE SPECIFICATIONS

The following specifications detail the project's expected measurements on technical performance. Both the technical aspects and artistic aspects of the game will be built and adjusted as needed in accordance with these specifications. If for some reason these specifications become unrealistic, they will be modified as necessary.

6.2.1 HARDWARE REQUIREMENTS

Though it will not be possible to accurately tell what the hardware requirements for *Blood Tide* will be until it is near completion, there are certain requirements that the C4 Engine already imposes.[5] The following is a list of these basic requirements:

- **Operating System:** Windows XP, Windows Vista, Mac OSX 10.5 or higher
- **Graphics Card:** Nvidia GeForce 6600 or higher, ATI Radeon X1300 or higher

The following is a list of the project's expected hardware requirements:

- **Operating System:** Windows XP/Vista
- **Graphics Card:** Nvidia GeForce 6600 or higher, ATI Radeon X1300 or higher
- **RAM:** 1 GB
- **Processor:** Intel Pentium 4 processor or better
- **Hard-disk Space:** 750 MB
- **Internet Connection:** 300kb/s or better

6.2.2 SOFTWARE REQUIREMENTS

Blood Tide will meet the following software requirements at minimum:

- **Players:** 2 players
- **Video frames per second:** 30
- **Units in world:** Maximum of 50

[4] Refer to http://www.terathon.com/forums/index.php/ (September 21, 2009)

[5] Refer to http://www.terathon.com/c4engine/faq.php (September 21, 2009)

6.3 TECHNICAL TOOLS

The project will be developed with Microsoft Visual Studio 2008. The C4 Engine by default comes packaged with the necessary visual studio project (solution) files to run, build, and debug the game. The team will be utilizing a WPI Sourceforge account to store all of the project's documents and source code. In order to access the source code, members will individually be using Tortoise SVN as the primary source-control software. Tortoise SVN was chosen because it is relatively easy to use and provides all of the functionality needed for source control, such as checking revisions to the code into the repository, checking modifications out to the local working copy, creating branches of the code-base for special needs, and providing a way to resolve conflicts with multiple code revisions.

6.4 TASKS

The team will be making use of GanntProject to create a task dependency chart (Appendix C), as well as Collabtive to act as a project management suite. Collabtive was chosen because it allows each member to log in to a website, see a list of tasks and when each task is due, and inform everyone else that a task is complete. This is a much needed tool for organizing which members get which tasks and when tasks need to be complete. All tasks will be assigned to an appropriate member of the group.

6.5 PLANNED COMPONENTS

The planned components section details all of the technical mechanisms that will be created or modified by the group in order for the game to work. In addition to explaining all of the components, this section will specify areas for a single person to work in, how important each component is, and the relationships and dataflow between the components.

The components are grouped into three categories: engine, interface, and gameplay (Figure 20). The engine components will provide basic functionality. The interface and gameplay components will make up the gameplay experience. These two categories will make use of the engine components and make use of each other, but the engine components will be independent of the other two categories.

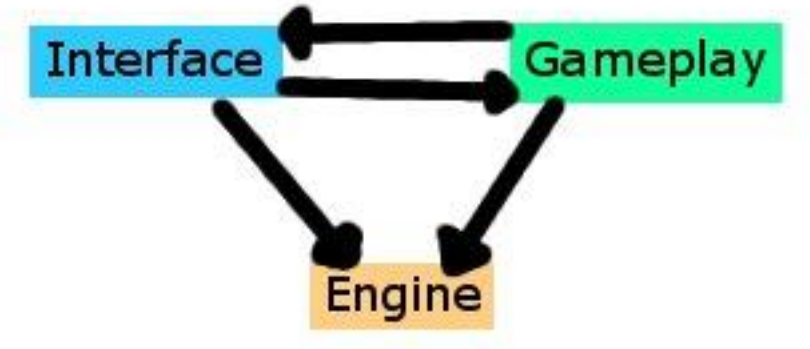


FIGURE 20: COMPONENTS

Due to the complexity of some of the components, they could fall into multiple categories but will be listed in only one. The importance of any given component will be described as core – the components that are basic functionality, required – the components that make up the gameplay, and desired – the components that improve the player experience but are not essential.

6.5.1 ENGINE COMPONENTS

The engine components consist of components that provide most of the underlying functionality for the game. These components are gameplay independent and will be mostly concerned with graphics, performance, mechanisms helpful to other components, and physics.

6.5.1.1 INITIAL SETUP

Initial setup will be composed of taking the latest build of the C4 Engine and removing sections of code that are not needed for this project. Since the tutorial sets up networking and game management, it is easier to remove a lot of code rather than to start from scratch and end up coding things that were already part of the tutorial. The goal of this component is to create a solid foundation of code that *Blood Tide* can be built upon.

This component is considered to be a core component.

6.5.1.2 ART PIPELINE

One of the major technical goals of the project is to make everything that is game-specific require no advanced coding knowledge. One of the major components of the game is to get the artwork (models and animations) into the game in a robust and scalable way. The team will be using text files, specifically string tables, to specify what characters and what models and animations are associated with it (Figure 21). This allows the team to be able to easily switch models, animations, and create new unit types very easily.

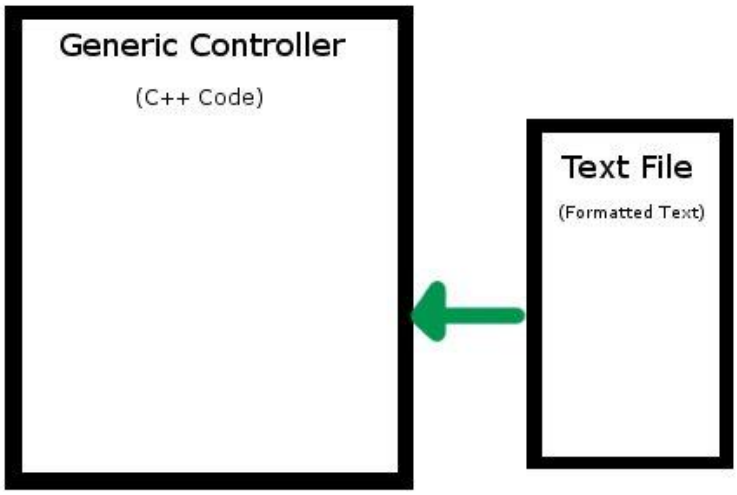


FIGURE 21: ART PIPELINE I

The art pipeline will be mostly used by entities within the game. The art pipeline will be utilized during the creation of entities as part of the initialization step (Figure 22).

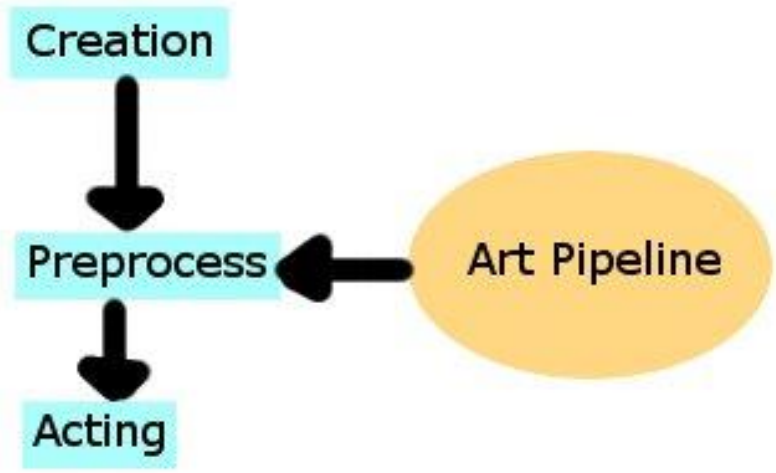


FIGURE 22: ART PIPELINE II

This component is considered to be a required component.

6.5.1.3 ZONING

Blood Tide will make use zones. This is one of the most important optimization features the team will be working on. Specifically, the game will have support for zones that will be automatically

generated instead of placed by hand, which makes level design simpler. The C4 Engine has a built in zoning construct that the team will be making use of.

Zones will be used in a variety of ways. First of all, they will be an important rendering optimization. Instead of possibly rendering every entity in the world, the game will only render entities in certain zones. Also, it will be much easier to figure out which entities are in a certain area of the map. As entities enter a zone, they will become part of that zone, so it will be very easy to get a list of each entity in a zone.

There won't be any new structures built for this component; instead, code to generate zones will be part of the map loading process. The code to correctly add entities to the zones will be part of the character controllers, specifically the unit controller and player controller.

This component is considered to be a required component.

6.5.1.4 NODE-GRAPH

Another major optimization feature will be the use of a node-graph. The team aims to have a graph of nodes automatically generated on the map. These nodes will be used primarily for path-finding. The node-graph component will also cache the results of ray tracing between the nodes so the path-finding algorithm will not have to do real-time ray tracing. The C4 Engine has an efficient pseudo-ray-tracing algorithm built into it that the team will be using.

The node-graph generation will work in two steps: node creation and point-of-view distance caching.

NODE CREATION

The first step in the node-graph process is to generate a series of nodes. This process will be part of the map loading process. An algorithm will run to create a grid of points, or nodes, on the map that are just above the ground. If a node is created underground, it will be pushed upwards until it is above the ground (Figure 23).

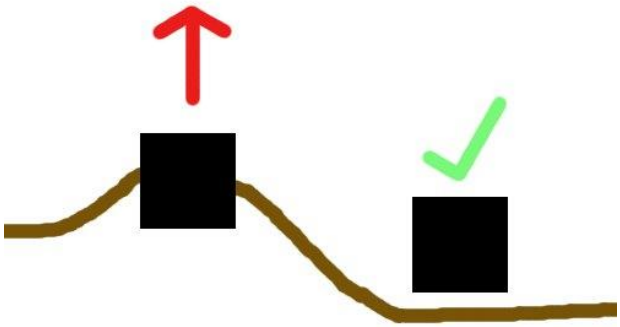


FIGURE 23: NODES

POINT-OF-VIEW DISTANCE CACHING

In order for a node-graph to work, the distance and whether or not other nodes are within some node's point-of-view must be calculated. First, the process will loop through every node and check the point-of-view with all of the other nodes. This will be done with the built in ray tracing algorithm that C4 comes with. Nodes that are not within the point-of-view of the current node will be discarded. After a list of nodes that can be seen is created, the next step is to calculate the distance to each of these nodes. A simple 2D distance calculation will be used. Later, when the path-finding algorithm needs to find the shortest path of nodes, these distances will be used. In order to make this process execute quickly, the results of these calculations will be cached so these values only need to be calculated once.

This component is considered to be a required component.

6.5.1.5 UNIT CONTROLLER

The unit controller component is the basic controller for each unit. This controller will be designed in a way such that it will have most of its information contained in string tables (as part of the art pipeline). The unit controller will be generic enough to work for all units. This controller will be based upon the player controller already built into the C4 Engine.

This component is considered to be a core component.

6.5.1.6 ARTIFICIAL INTELLIGENCE

The artificial intelligence component consists of three other components: movement, path-finding, and unit attacks. This component will be built as a standalone AI mechanism. Unit controllers will then be able to use any AI mechanism they choose (Figure 24: Unit AI Plug-in). This allows the team to create a variety of AI plug-ins to exhibit specialized or different behavior easily.

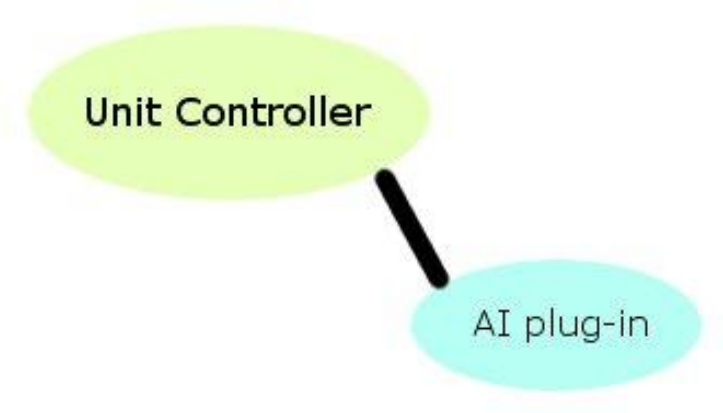


FIGURE 24: UNIT AI PLUG-IN

MOVEMENT

This component is for units and unit AI. Movement consists of goal-generation for a unit, steering mechanisms to choose a heading, and locomotion algorithms to actually move a unit throughout the map (Figure 25).

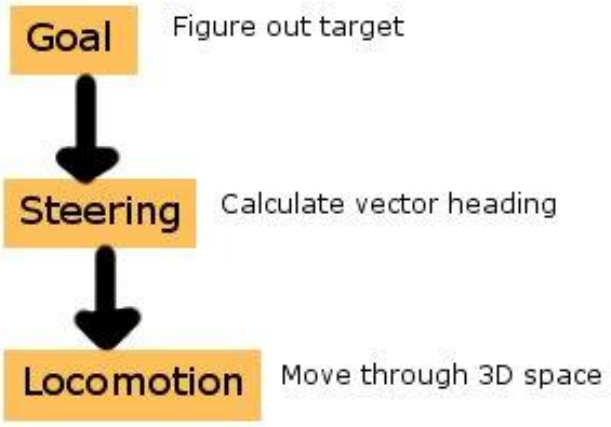


FIGURE 25: MOVEMENT HIERARCHY

Goal generation ultimately picks a target, specifically some vector in the game space. Whether the entity is moving towards a location or towards another entity, there is a vector location that it is moving to. Goal generation will also keep track of whether or not the unit wants to stop, or have no target at all. In order to get to a target, a unit must either go directly there, or pick a certain path if there are obstacles in the way. Only static obstacles will be handled by path-finding.

Steering is the process of calculating a new vector heading for the entity. Instead of just facing the target, steering should slowly move to a target to make entities move and turn smoothly (Figure 26).

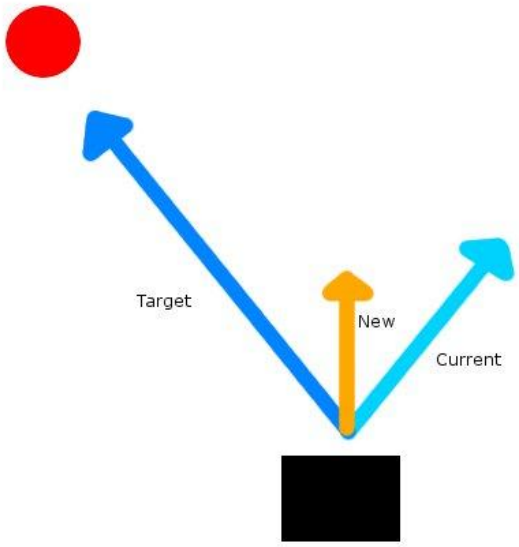


FIGURE 26: UNIT LOCOMOTION

Locomotion is the process of changing the entity's position in space. This process will take into consideration velocity, acceleration, and smoothing techniques. As part of the locomotion layer, there will be collision detection. Entities will have collision detection with the environment and with other entities in the game world. Whereas path-finding is for static obstacles in the game world, collision detection will be used for dynamic obstacle avoidance.

This component is considered to be a core component.

PATH-FINDING

Path-finding shall consist of generating a series of waypoints, which will come from the node-graph component, in order for a unit to move around the map without getting stuck on obstacles. The plan is to use an A* search algorithm in conjunction with the node-graph component to make path-finding real-time and efficient.

The first step in the path-finding process is to check if the entity that should move can see the target. If it can, there are no obstructions and the entity should proceed to the target. If there are obstructions, the entity will need to find a node that can both see the target and the entity. Since there could be multiple nodes that fit this requirement, the node with the shortest total distance (entity to target) will be chosen. It is possible that a series of nodes will need to be chosen. If there is absolutely no path that can be made from the entity to the target, the entity will simply not move.

This component is considered to be a required component.

UNIT ATTACKS

Unit attacks consist of a unit realizing it is able to attack an enemy and carrying out the action of attacking. To carry out the action, the unit will have to play an attacking animation, calculate whether or not it damaged the enemy, and then to inform the players and the enemy that damage has been dealt.

This component is considered to be a core component.

6.5.1.7 FOG OF WAR

The fog of war component hides or displays the opposing factions units. Every building and unit will keep track of what it can see. From these lists, an overall can-see list will be kept that will be used to actually display or hide entities in the world.

This component is considered to be a desired component.

6.5.2 INTERFACE COMPONENTS

The interface components create a way for players to interact with the game. These components are concerned with controls, 2D user interface, cameras, and display of information.

6.5.2.1 RTS CONTROLS

The RTS controls component will control user input while in command mode. This includes functionality for the player to move, zoom, and rotate the camera. It also supports the selecting game entities and interacting with the RTS UI.

The controls are mostly built into the C4 Engine. In order to make the controls work, various portions of the control scheme will be modified to act like an RTS game. The RTS controls will have separate key bindings that can be loaded or unloaded at any time from other control schemes.

This component is considered to be a core component.

6.5.2.2 ACTION CONTROLS

The action controls component will control user input while in field mode. This includes functionality for the player to move, look around, and use abilities.

Like the RTS controls, the basic C4 controls will be modified to act like an action game. The key bindings for this scheme will be kept separate from other control schemes.

This component is considered to be a core component.

6.5.2.3 PLAYER CONTROLLER

The player controller component consists of creating a player controller for the field-commander. It will handle user-input and other messages that affect the field-commander and update its state, model and animations accordingly. The player controller will be generic and support each type of field-commander.

This component is considered to be a core component.

6.5.2.4 GAME LOBBY

The game lobby component consists of a lobby screen for the two players to use before starting a game session. It will allow the player's to select which faction to play as.

The game lobby will keep track of which players are on which team, including players who have not yet chosen a team. It will also provide support for an interface for players to choose the various options that the lobby supports.

This component is considered to be a required component.

6.5.2.5 RTS UI

The RTS UI component will provide the player with information pertaining to units, such as the orders that can be issued and the stats and state of units. The basic components for making the UI are already built into the C4 Engine. These will be used as building blocks to create the RTS UI.

This component is considered to be a required component.

6.5.2.6 ACTION UI

The action UI component will provide the player with information pertaining to the field-commander, such as the field-commander's health, stats and rank. The basic components for making the UI are already built into the C4 Engine. These will be used as building blocks to create the action UI.

This component is considered to be a required component.

6.5.2.7 BUILDING UI

The building UI component consists of a pop-up UI element that displays the building choices for a plot of land. The basic components for making the UI are already built into the C4 Engine. These will be used as building blocks to create the building UI.

This component is considered to be a required component.

6.5.2.8 MENU SYSTEM

The C4 Engine has a built in menu system. This base menu system will be modified to display only relevant information for *Blood Tide*. The new menu system will also contain the art team's menu art.

The component is considered to be a desired component.

6.5.3 GAMEPLAY COMPONENTS

The gameplay components create the specific gameplay mechanisms needed to have rules, goals, and provide a flow of game events.

6.5.3.1 STATS CONTROLLER

The stats controller stores all of the stats for a unit or building. This component will support an easy-to-use interface for adjusting the stats of a given unit or building. The controller will be generic and act like a plug-in for entities that need to keep track of their stats (Figure 27).

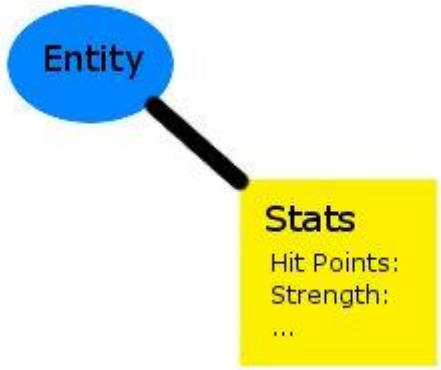


FIGURE 27: STATS CONTROLLER

This component is considered to be a required component.

6.5.3.2 COMMANDER SPAWN

The commander spawn component controls where and when the field-commander will spawn on the map. It will also keep track of time periods when a field-commander is not allowed to spawn, such as the time after a field-commander dies.

This component is considered to be a required component.

6.5.3.3 BASE CONTROLLER

The base controller component is the controller of each faction's bases. It will be responsible for the status of each base, as well as the spawning of the corresponding units at the base. The base controller will be connected to a series of spawn points and plot controllers.

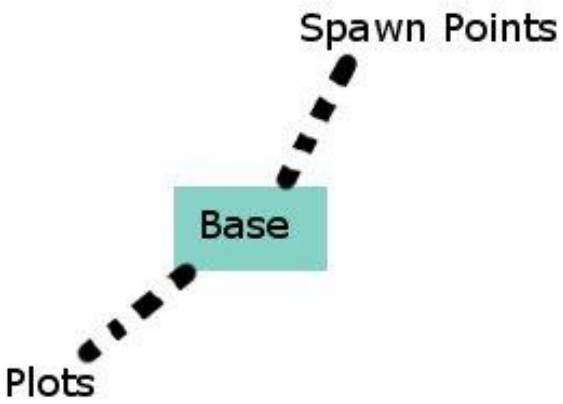


FIGURE 28: BASE CONTROLLER

This component is considered to be a required component.

6.5.3.4 PLOT CONTROLLER

The plot controller component controls each plot that surrounds a base. It keeps track of what building is built upon it and acts as a mediator between buildings and bases. A plot controller is connected to some base controller (Figure 29).

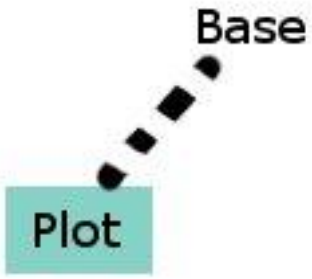


FIGURE 29: PLOT CONTROLLER

This component is considered to be a required component.

6.5.3.5 VICTORY CONDITION

The victory condition component provides customizable player goals in a game session. This component will be built as a generic controller that can specify any number of victory conditions as gameplay dictates. The controller will be placed via the world editor.

This component is considered to be a required component.

6.5.3.6 WORLD EDITOR TOOLKIT

The world editor toolkit allows bases, plots, buildings, victory mechanisms, and other gameplay-specific mechanisms to be placed and created in the World Editor, so that maps can be easily created.

This component is considered to be a desired component.

6.5.3.7 SPECIAL EFFECTS

The special effects component controls the creation and display of special effects including particle effects and camera blending.

PARTICLE EFFECTS

There are already particle effects for things like fire built into the C4 Engine. For particle effects that resemble bubbles, the basic particle effects will be modified. Other particle effects will include effects seen on attacks, such as explosions, bombs, and debris.

CAMERA BLENDING

Camera blending is mostly part of the switching between field mode and command mode. Instead of the camera switching instantly from one to the other, the camera will move over time to the new position. For blending, there will be a timer that has a set interval and length. Over time, the camera's position, angle, and heading will slowly change to become the new camera (Figure 30).

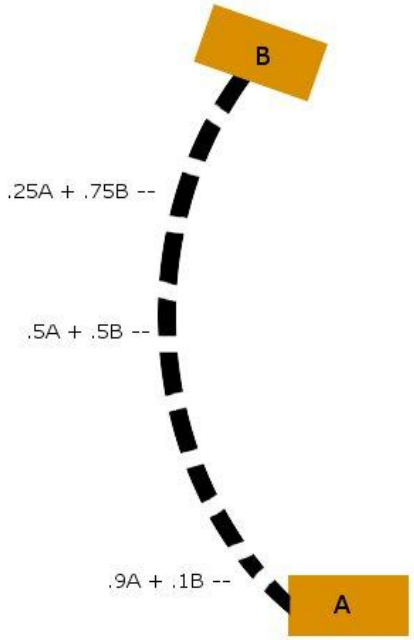


FIGURE 30: CAMERA BLENDING

UNIT FADING

In the fog of war system, units appear and disappear. Instead of this being a binary action, units will fade in or out. Much like the camera blending effect, the opacity of units will increase or decrease over time.

This component is considered to be a desired component.

SHADERS

Custom shaders will be built to simulate underwater environmental effects. These will be similar to blurring effects.

6.5.3.8 SOUND EFFECTS

The C4 Engine has a built in sound manager that the team will be modifying. For background music and generic sounds, the built in mechanisms will be used. For sounds that are location specific, the team will be using a sound source mechanism, also found in the C4 Engine. The sound source will be attached to the entity that generates the sound (Figure 31). If the entity moves, so too will the sound source.

In addition to attaching sounds to entities, the process of deciding which sounds to play during various actions will be handled dynamically. As part of the art pipeline, sounds will be placed in

text files and will specify when they should play. Sounds will also specify whether or not they are local or global sounds.

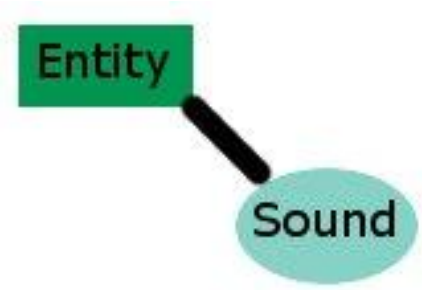


FIGURE 31: SOUND EFFECTS

This component is considered to be a desired component.

6.5.3.9 HINT SYSTEM

The hint system will provide textual and auditory hints for players to learn the game. This component will be built in conjunction with the artists' work for an in-game tutorial system.

This component is considered to be a desired component.

6.6 PROJECT MILESTONES

During the implementation and testing phases of this game, the team will have regular meetings to discuss the progress of the game and create plans for what needs to get done to fix problems or continue development. There will be many builds of this game, but there are clear milestones that the team will be working towards. What follows is a list of these milestones:

- **First Playable –** The first functional build of the game. This build will display some basic functionality and will have placeholder art. The components expected to be completed are: initial setup, art pipeline, zoning, node-graphs, unit controller, unit movement, RTS controls, action controls, player controller, game lobby, unit stats, action spawning, base controller, and plot controller. This milestone is scheduled for December 10, 2009 (sixth week of B term).

- **Alpha –** The second functional prototype. This build will showcase most of the main components being built for this game. Specifically, the components expected to be finished for this milestone are: unit path-finding, unit attacks, fog of war, RTS UI, action UI, building UI, dynamic buildings, victory condition controller, and unit healing. This milestone is scheduled for January 29, 2010 (third week of C term).

- **Beta –** The first public prototype. This version will be feature complete and will have most of the art content. This milestone is scheduled for March 4, 2010 (last week of C term).

- **Gold –** The final public release as part of this MQP. This will be feature complete, have all artwork in place, be thoroughly debugged and play-tested, and will be publicly available for download. This milestone is scheduled for May 4, 2010 (end of D term).

6.7 PROJECT DEPLOYMENT

Once complete, the game will be made publicly available for download on the game's website[6]. Also, the team will be posting updates, screenshots, videos, descriptions of the game, and other content on the website. The game will be released a single file download. This file will be a setup script that will automatically install the game. This will allow anyone who wants to play the game to be able to easily obtain a copy of the game. The team will be using the Nullsoft Scriptable Install System[7] to create the installation script.

[6] Refer to http://bloodtide.demalus.com/

[7] Refer to http://nsis.sourceforge.net/Main_Page (October 4, 2009)

APPENDIX A: ART ASSET LIST

TABLE 6: MODELS AND TEXTURES ASSET LIST

Models & Textures		
Units:	Merman Commander Texture	Will be used with Merman unit model
	Scuba Commander Texture	Will be used with Scuba Troop unit model
	Turtle Commander Texture	Will be used with Turtle unit model
	Submarine Commander Texture	Will be used with Submarine unit model
	Manta Ray Commander Texture	Will be used with Manta Ray unit model
	Bomb Ship Commander Texture	Will be used with Bomb Ship unit model
	Mermen Unit	
	Scuba Troop Unit	
	Turtle Unit	
	Submarine Unit	
	Manta Ray Unit	
	Bomb Ship Unit	
Buildings:	Mermen Base	
	Turtle Base	
	Manta Ray Base	
	Piscivian Farm	
	Piscivian Training	
	Piscivian Guard Tower	

Environment:	Scuba Diver Base
	Submarine Base
	Bomb Ship Base
	Human Farm
	Human Training
	Human Guard Tower
	Destroyed Base/Building
	Empty Plot
	Coral Reef
	Sea Weed
	School of Fish
	Rocks
	Sea Shells
	Sea Sponge
	Sunken Subs(WWII)
	Unexploded Bombs (WWII)
	Ruins
	Starfish
	Jelly Fish

TABLE 7: ANIMATIONS ASSET LIST

Animations	
Merman	Attack 1
	Attack 2
	Attack 3
	Move
	Death
Scuba Troop	Attack 1
	Attack 2
	Attack 3
	Move
	Death
Turtle	Attack 1
	Attack 2
	Attack 3
	Move
	Death
Submarine	Attack 1
	Attack 2
	Attack 3
	Move
	Death

Manta Ray	Attack 1
	Attack 2
	Attack 3
	Move
	Death
Bomb Ship	Attack 1
	Attack 2
	Attack 3
	Move
	Death

TABLE 8: MUSIC ASSET LIST

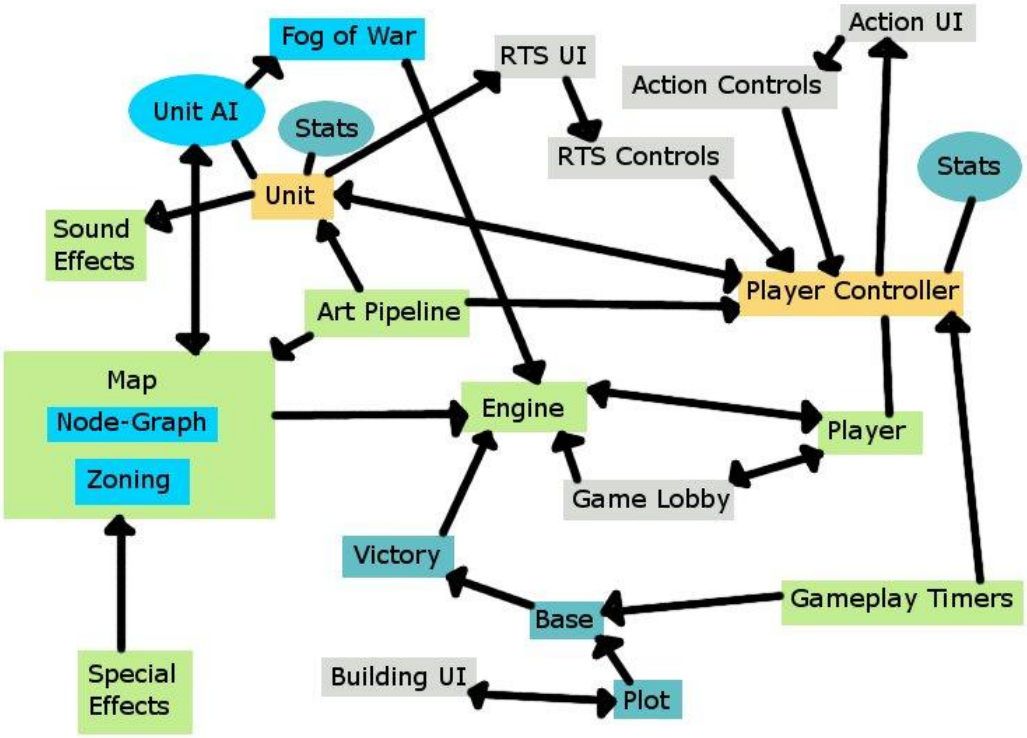
Music	
Song	Menu Song
	Command Mode Song
	Field Mode Song
SFX	Organic Unit Death
	Mechanical Unit Death
	Building Created
	Building Destroyed
	Unit Spawn
	Field-Commander Spawn
	Merman Attack
	Merman Move
	Merman Accept Orders
	Scuba Troop Attack
	Scuba Move
	Scuba Accept Orders
	Turtle Attack
	Turtle Move
	Turtle Accept Orders
	Submarine Attack
	Sub Move

	Sub Accept Orders
	Manta Ray Attack
	Manta Move
	Manta Accept Orders
	Bomb Ship Attack
	Bomb Move
	Bomb Accept Orders
	Turret Tower Attack
	Jelly Tower Attack

APPENDIX B: ART TIMELINE

	Joseph Alea	Sarah Gilkey	Sean Beck
Week 1	Merman, Scuba Base	Scuba Troop, Merman base	Submarine, Sub Base, Sub Animations
Week 2	Animations: Scubatroop, Merman Commander Tex.	Animations: Merman, Scuba Commander Tex.	Manta Ray, Manta Base
Week 3	Turtle and Base, Human Guard	Animations: Manta Ray, Pisc. Guard Tower, Empty Plot	Bombship & Bombship Base
Week 4	Animations: Turtle & Bombship	Remaining environment, Destroyed Base, Map	Human Farm & Univ, Pisc. Farm & Univ
Week 5	Menu art - human	Menu art - piscivian	Sub Commander Tex, Manta Commander Tex, Bombship Commander Tex
Week 6	In Game music, SFX - sub, turtle	Main menu music, SFX - manta, bomb	SFX - merman, scuba troop, general, buildings
Week 7	Finish Remaining Assets	Finish Remaining Assets	Finish Remaining Assets

APPENDIX C: DATA FLOW



APPENDIX D: TECHNICAL TIMELINE