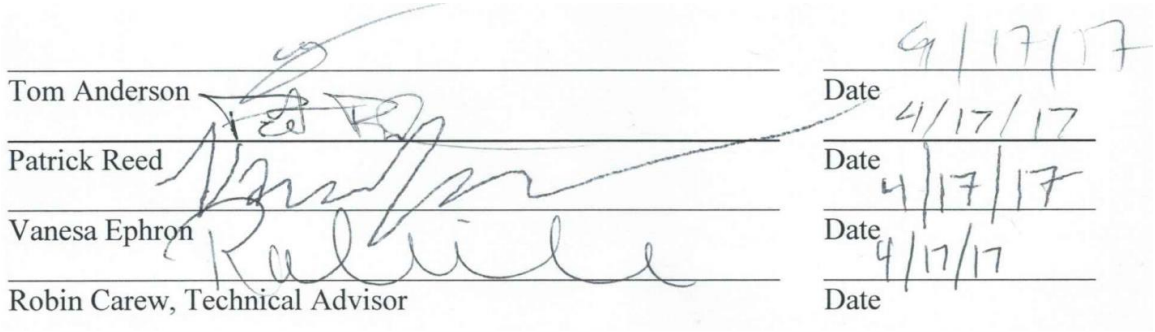
Blu3_M1sT

by Team 1

Tom Anderson, Patrick Reed, Vanesa Ephron

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology



University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2017

Table of Contents

ABSTRACT

Blue Mist is a dystopian cyberpunk tactics role-playing video game (RPG) designed to tap into the market which demands a more well-rounded dynamic world from the turn-based Computer RPG genre. As today's major game studios widens their appeal, we've created Blue Mist from the ground up to give the genre's fan base the return to basics they always wanted. In Unity 3D, the game features all the classic elements of cRPGs with turn-based tactical combat, skill-based character progression, and a story driven experience. This is enhanced with the unique neon-filled cyberpunk setting, a hand-crafted cast of characters, and an adapting world which morphs to reflect the choices of the player. We have also incorporated an innovative event system which smartly dictates the flow of the game, in order to make every playthrough a fresh experience and guaranteeing replayability.

INTRODUCTION

The video game industry had always been one to reward eccentric creativity, unyielding passion, and a desire to provide for customers of all backgrounds. As it aged, a wide array of new genres surfaced to match the demand of potential customers. In recent years, mobile gaming had seized a large and previously inaccessible market. This trend had drawn a lot attention toward the casual gaming audience, who played on their mobile devices and were accustomed to the business model of micro-transaction. This left a hole in the supply of core gamers, those who played on consoles and computers, for a rich and streamlined gameplay experience which we stepped in to fill.

PROBLEM

The AAA gaming industry primarily focused on developing franchises and selling downloadable contents (DLCs). While this appealed to most of their player base, there were still a lot of gamers out there with demand of more focus, niche, and innovative titles. There had been a few products advertised to the crowd who wanted a tactical role-playing game (RPG), but their unique strengths and weaknesses left customers picking between either a rich storytelling experience or strong mechanics and replayability, and nothing which would satisfy both.

SOLUTION

The solution to a blank space in the market was to develop a suitable product that would fit it in. We set out to create Blue Mist as a tactical squad-based RPG at its core, while also featuring procedurally generated missions, campaign map strategy, handcrafted characters, and custom event system. This aimed at providing both a strong narrative to explore, as well as solid mechanics for those who preferred a more straightforward experience.

PROJECT DESCRIPTION

Blue Mist is a game made in Unity3D, set in the post-nuclear cyberpunk United States of 2060, where what's left of the population has holed themselves within dystopian cities ruled by rogue nations, leaving behind a barren radioactive wasteland. The game's goal is to strike a delicate balance between small-scale mission-based micromanagement, and the large-scale map-based conquest campaign to build a solid base for gameplay. This, supported by the thematic setting and diverse cast of three-dimensional characters, has the potential to be a trendsetter in a volatile market.

OVERVIEW

The rest of the document explains the planning and development phases of the project, with their respective requirements. This encompasses project concept, user profiles, objectives, methodology, budget, timeline, discussions, and conclusion.

DISCUSSION

PROJECT CONCEPT

The concept for the game originated as a small idea that was meant to be a reimagining of the Oregon Trail concept, where the player took on a team to survive the harsh setting of an apocalyptic dystopian world. From then, a mix of market research and team member suggestions removed the survival elements in hopes of making this a bigger project, citing parameters such as thirst and hunger served as only a numeric distraction in a campaign. This was when the possibility of combat was put on the table, and we settled on turn-based combat as the most immersive form of progression while also allowing the player to soak in the world and characters at their leisure and discretion.

PROJECT DESIGN

The development of the game was broken into several major pieces, which were then distributed to team members based on ability and availability.

1. Core systems:

In a tactical game, the core systems were the most important aspect of the experience. It would be what the players spend most of their time doing, and was where we focus most of our development time. In Blue Mist, this is divided into mapping, movement, and combat, which would come into focus during a mission, and expected player's input for unit micromanagement.

The map on which the missions takes place are all based on real landmarks of the United States, though modified to suit the time and context. A square grid is then layered on top of it to allow for unit movement.

The player controls a squad of units to whom he or she will give orders. The squad contains the player's own avatar, as well as whoever the player takes with him or her on the mission. Units are allotted action points based on their unique characteristics, which the player can spend to move, engage in combat, or interact with the environment. Here, the player is confronted with enemies, traps, and puzzles, through which they must battle or outsmart to advance.

Combat is a ranged cover system. Each unit had their own hit points, which represented their vitality. A side wins when they deplete all their enemies' hit points. Units controlled by the player can attack with their weapons, use abilities, or activate items to gain the upper hand in combat and come out victorious. If the player's squad is wiped out at any point, the mission will fail.

2. RPG elements:

The RPG elements sits at the heart of gameplay. Each unit in combat is derived from characters which the player can either customize or recruit to their liking. Each character features their own set of abilities (actions which they could use during battles), attributes (passive statistics which defined their physique), and backstory (which gave them depth and personalities). Characters also can equip items, which further prepares them for missions ahead. Here, the character's attributes determine how well they could perform actions during combat,

and their items would either enhance these attributes or grant them more abilities. Characters also have loyalty, which the player can influence through their diplomacy and actions. See 'management mode' below for details.

3. Management mode:

Blue Mist's management gameplay consists of a campaign-style map conquest. The player begins with control of a small base. From here they can grow their influence through recruitments of characters, buying of items, researching of upgrades, and participating in politics. They do this by sending a squad on missions, the success of which will reward them with currencies and territory.

Characters can be recruited through events (see 'auxiliary system' below), claiming of territory as a reward, or quests. Some characters cost currencies to recruit, or require pay, some asks for neither. A character's loyalty will affect his or her chance to defect, which the player can influence by completing quests. High loyalty characters may also give the player unique missions and rewards.

Currency can be acquired through mission rewards or completion of quest. Items can be bought through vendors travelling through controlled territories, or through trading with other factions which would incur a tax. Trading costs currencies and sometimes transportation time.

Upgrades improve the squad's effectiveness in combat by giving them better equipment, or providing them with the ability to train and improve their attributes. Upgrades costs currencies and time.

The game features several other factions with their own motives. Each faction represents a rogue nation, with their own territory and army, with which the player could do politics. Certain actions the player take (sending squads on missions, capturing territories etc.) will affect different factions in different ways. The general behavior of a faction directly influences their aggressive or passiveness toward the player. The player can form alliances with factions given a positive enough opinion.

The player beats a playthrough when they or their alliance control all possible territory on the map. The player loses the game when the player avatar dies, or he or she loses control of all territories.

4. Auxiliary systems:

The game features a smart random system dubbed 'controlled chaos'. This system's goal is to determine the randomness of the events in the game. Events have different effects with different varieties, and the system is in place to guarantee that an extreme string of bad luck would not make it impossible for a player to progress. It will also make sure to keep the gameplay fresh by not repeating random events, as well as persisting these settings through different playthroughs to make them as unique as possible.

The game also features an organic world. Based on which mission the player elected to pursue, and how they pursue it, the game world will dynamically adapt to the result of his or her actions. For example, if the player assassinates a higher-up figure of a neutral faction, this will lead to a drop in their influence toward said faction. This will in turn raise the trading tax, as well as lower the loyalty of any follower who favored said faction. This will also make the

faction more likely to attack the player in retaliation, creating a cascade of effects that weighed on the player's choices.

5. Content and balance:

Content in this paper is defined as the collection of characters, items, events, and factions, which directly impact the player's experience, and constantly adds in to improve the gameplay variety. Each piece of content requires their own visual representation (see 'graphics' below) and sound effects (see 'audio' below). The more content the game has, the smaller possibility that any two playthroughs will be the same, aims to make sure the player received a fresh run.

With more content added to the game, they will need to be properly balanced. The aim of balance is to make sure that while each character, item, faction is unique in their own way, they should not be so strong, cheap, or difficult that would influence player's choice more than just their sense of progression and playstyle. This means that every piece of content, if available, should present a meaningful reason for the player to access it. Characters are balanced around the strength of their abilities, the ease of accessing them, the cost to hire them, as well as their attributes. Items are balanced around their effects, cost, and scarcity. Factions are balanced around their behavior, military strength, and territorial control.

Events act differently, as they would never be the player's direct choice, instead chosen for them by the game system. Therefore, they do not need to be balanced in the same way as other pieces of content. Events have effects which ranges from very bad to very good, and will occur over a period, or directly as the aftermath of a mission, over a varying scale. Events with stronger or more significant effect have a lower chance of occurring, and the controlled chaos system

makes sure that they didn't happen too often to make sure the player doesn't feel cheated due to a series of bad luck.

6. User Interface (UI) & Experience:

The game's user interface must fulfil these basic principles: give the players exactly what he or she wants immediately, or get out of the way (Quintans 2013). This was further expanded to the following points:

- Important statistics must be constantly present. This includes abilities and their availability in combat, as well as character's health and actions.
- It should take users as few clicks as possible to get to more in-depth information should they require. Things like inventory, character statistics, explanations of abilities should always be one click or hover away.
- The interface should be self-explanatory. Statistics must be attached to their specific characters, and abilities with similar conditions and functions must be grouped together.
- Highlight available actions for the turn, and dim them otherwise. If a character can move, how far he or she could move should be shown. Abilities not ready to use should not light up.
- No animations. Animations are made for characters, not the interface. The interface should get the user exactly where he or she wanted to be as fast as possible. Possibly very short delays may be implemented to prevent jarring transitions.

- All actions must have shortcuts that would be shown always. Access to a keyboard should help the player.

7. Graphics:

All content in the game and the game itself requires graphical presentation. The game's art direction is designed to closely follow the cyberpunk dystopian theme. This means sharp modern hardline designs, and minimalistic flat UI elements. They are divided into different groups:

- Character concept arts: 2D sketches for character designs.
- 3D modelling: in-game implementations of the concept arts
- Icons: to represent items, events, UI actions, and abilities
- Maps: to represent the movement grid layers as well as the management mod themes
- Logo: for branding purposes of the game

8. Audio:

Audio is implemented to supplement the player's immersion. Actions should play sound effects along with animations to indicate their taking place. The game also features music with varying degrees of tempo to match the situation and the game's theme. They slow down during the management phase and free movement, and speed up during high octane combat.

UNITY

The Unity3D engine was the only significant piece of technology which we utilized throughout the process. Everything from scripting and models to animations and UI are done within the engine's Graphical User Interface.

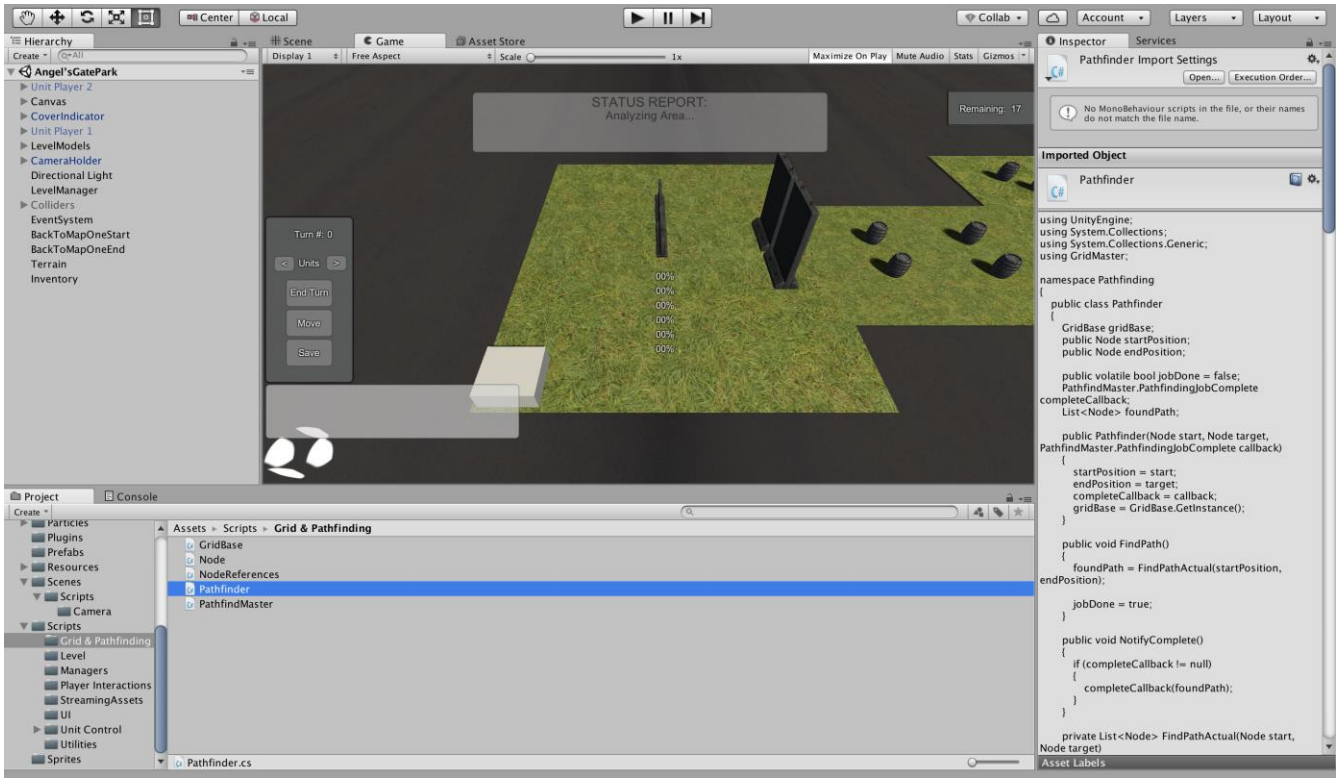


Figure 1: Unity User Interface

This was adequate for our purposes, as an all-in-one solution for both front-end and back-end work made sense for our process, allowing us to see the immediate changes of what we have made in the code. The fact that its visual-based means that accessing assets already present on

screen will also be quicker and more precise.

Figure 2: Work-In-Progress Main Menu

The engine itself gives enough tools to make a rudimentary interface for the game without requiring any extra art assets, something to which we didn't have access (GameGrind 2015).

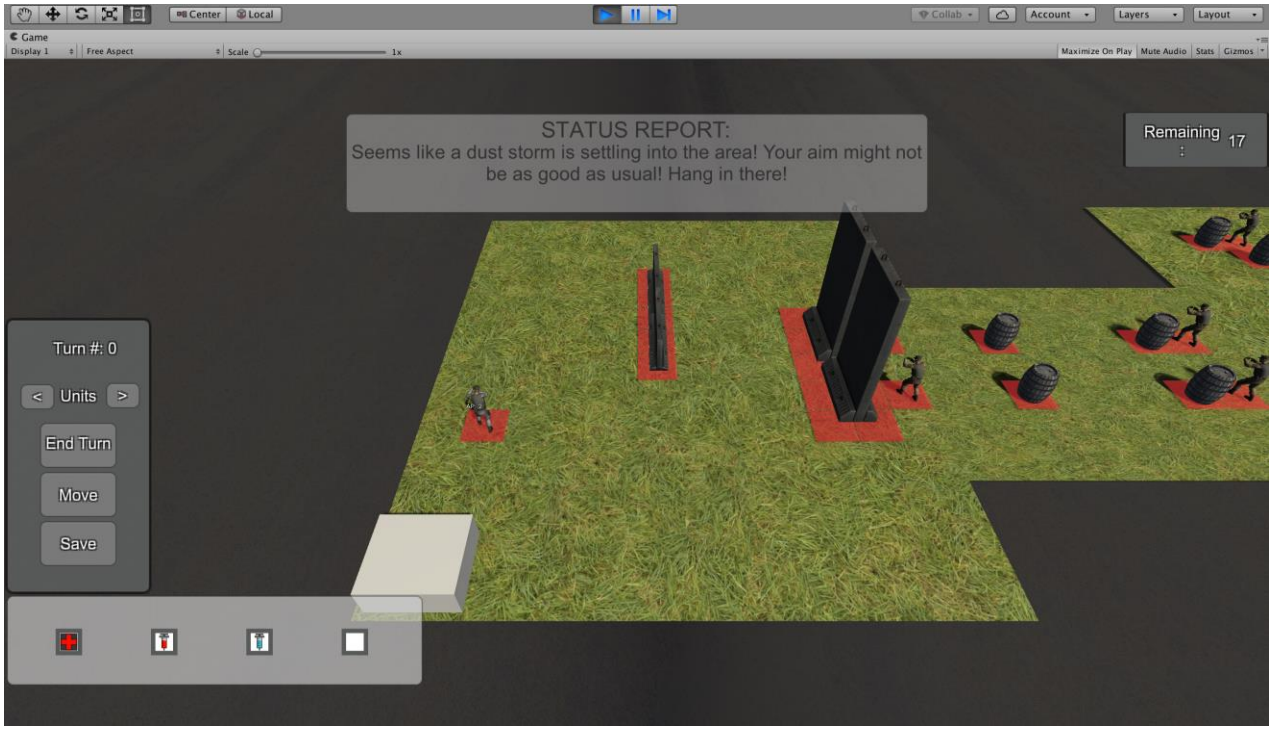


Figure 3: Play testing

The engine also allows us to test every change we implemented immediately after they're committed to the code base. This holistic bundle approach avoided the common pitfalls seen in processed software development, which is a slow and often buggy process of connecting files.

Unity also provided unique problems. Being such an all-purpose engine, there were a lot of specific configurations which needed to be done in order to tailor the engine to suit our needs. The constant updates to the engine's own code base was also something that had interfered with the synchronization process at times. The three of us operated on sometimes three different versions of Unity, each running potentially deprecated methods causing unforeseen problems. This also made version control extremely difficult, since the usual solutions of Git were never meant to handle large asset files and updates which accompanied game development.

Regardless, Unity was the best and only choice of a game engine. It didn't have all of the specific necessities we required, but developing over small hurdles like that was always possible. Furthermore, its familiarity and popularity with the game development community at large meant that we had a lot of support from peers as well as ready-made tutorials, although the frequent version updates of the engine rendered a lot of them unreliable over the long run.

USER PROFILES

1. Players:

Players are the primary consumers of our product. The first goal of development has always been to provide the players with an enjoyable gaming experienced, achieved through a combination of accessibility in user interface design, low barrier of entry, and usage of familiar and interesting themes. The desired playtime for players are three to five hours stretches, with an emphasis on the drop-off and pick-up-where-you-left-off philosophy.

2. Unity:

Unity is the platform with which we have decided to develop our game, primarily scripted in C#. Compliance with Unity's standards as well as taking advantage of its rich third-party integration is a top priority. The game is developed primary for the Windows Operating System, with the possibility to port to different platforms.

3. Publishers:

Publishers is an important aspect of our success. They provide funding, coverage, and other support for the game and its release. Extra funding helps us purchase additional assets to speed up production of the game in systems which has already been done before. Coverage helps us get an early and meaningful gauge of the market's reaction to our eventual release. Support helps us avoid common pitfalls in the process of development, release, and maintenance.

4. Distributors:

Cooperation with distributors such as Valve and CD Projekt will be a very important step in getting our game on their respective platforms of Steam and GOG, through which it could reach the player market. The popularity and promotion of these services is well worth the inevitable distribution cut.

5. Journalists:

Gaming journalists and media, traditional or video-based, play another key role in the marketing of our eventual release. They are important to generate interests before the final product was released, and act as another news platform.

USE CASE DIAGRAM

Below is the use case diagram for our video game product, showing the interlaced interactions with different stakeholders help us sell, market, and distribute our game, while also giving us feedback to make it better.

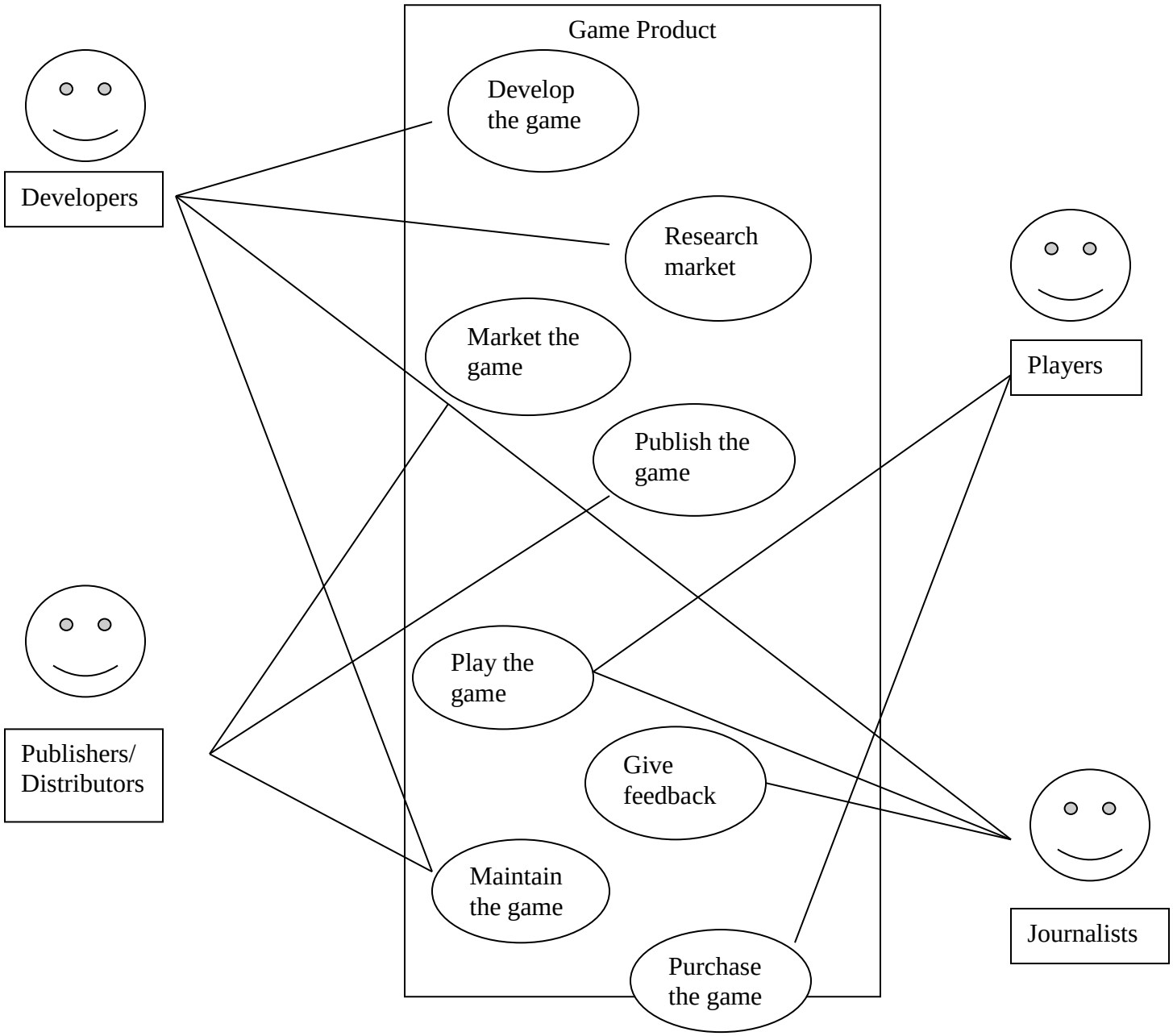


Figure 4: Use Case Diagram

METHODOLOGY

We have been employing a basic software release life cycle to comply with market standards and what our customers will expect. The product is developed and maintained from the initial concept phase until the maintenance phase, divided into two periods: development and release.

The development period consists of pre-alphas, alphas, betas, and release candidates.

Pre-alpha is where we discussed the product concept, scope, the approach we were going to take to solve each problem, and the timeline to which we would stick to finish the features. The basic prototype is developed in this phase and used as a piece of presentation as well as pitch for publishers.

Alphas would be versions where we introduce different features into the product, along with the testing of their functionalities. Alpha builds will continue until all planned features are implemented into the product. Beta versions would then follow, where our primarily focus is on bug-testing and rounding out the graphical and audio assets for the final product (Farlex n.d.).

Release candidates is when the product is due to go through a final testing phase, where it will be ready to release as-is unless a significant bug is found. The game at this point will be considered feature completed, and thoroughly tested. During the release period, we will roll out our code and assets to various distribution platforms, where the product is made available to the public. From then, our focus will shift toward releasing potential bug fixes, as well as adding new features if demanded. These new features compound to another development cycle as described above, and finish at another distribution platform rollout.

	Scenario 1	**Scenario 2**	**Scenario 3**	**Scenario 4**
Description	Simple Combat	Screen Transition	UI Elements	Event Calculations
Member	Patrick	Patrick	Tom	Vanesa
Area				
Graphic	X	X	X	
Model	X			
UI		X	X	X
Controls	X		X	X
Calculations	X			X
Variables	X		X	X

Table 1. Blue Mist's Testing Scenarios Tracker

BUDGET

Unity3D is a free software for us in Team 1 as startup independent developers. Their marketplace features scripts and packages for money. Currently we do not have any plan to use them, but may elect to in the future.

As the plan for the project is to go commercial, the budget proposal contains all possible expenses required to complete the project as was. We currently use all personal equipment for the development of the game, but this would also be included in the budget. Labor cost was

calculated at $20/hour, with the current rate of 15 hours each person per week. The budget estimate included everything from the conception to the future release of this game as a commercial product.

No.	Name	Count	Price	Total
1	Computer Hardware	6	$600	$3600
2	Adobe Fuse CC	1	$20/month (estimated 1 year)	$240
3	Labor	6	$20/hour (estimated 36 weeks)	$64,800
4	Unity3D	6	$0	$0
Estimated Subtotal				$68,640
Actual Total (No labor cost)				$3840

Table 2. Blue Mist's Estimated Budget

GANTT CHART

The following Gantt Charts are meant to serve as a timeline estimates for implementations of features into the product pre-release. It includes items which is completed by either the core team or the support team, and their expected finish date. The core dungeon systems are our priority, since they are necessary for our first demo and playtesting. Management systems will be implemented to expand the effects of the core systems. Content will trickle in as needed to provide variety while remaining balanced.

Task/Week	3	4	5	6	7	8	9	10	11	12
Game Design										
• Scope										
• Features										
• Content Creation										
• Plot & Setting										
• Balance										
Dungeon Systems										
• Grid Creation										
• Character Render										
• Movement										
• Map Design										
• Combat										
• Cover										
Management Systems										
• Inventory										
• Resources										
• Characters										
• Events										
• Research										
• Politics										
• Missions										
UI										
• Main Menu										
• Transitions										
• Maps										

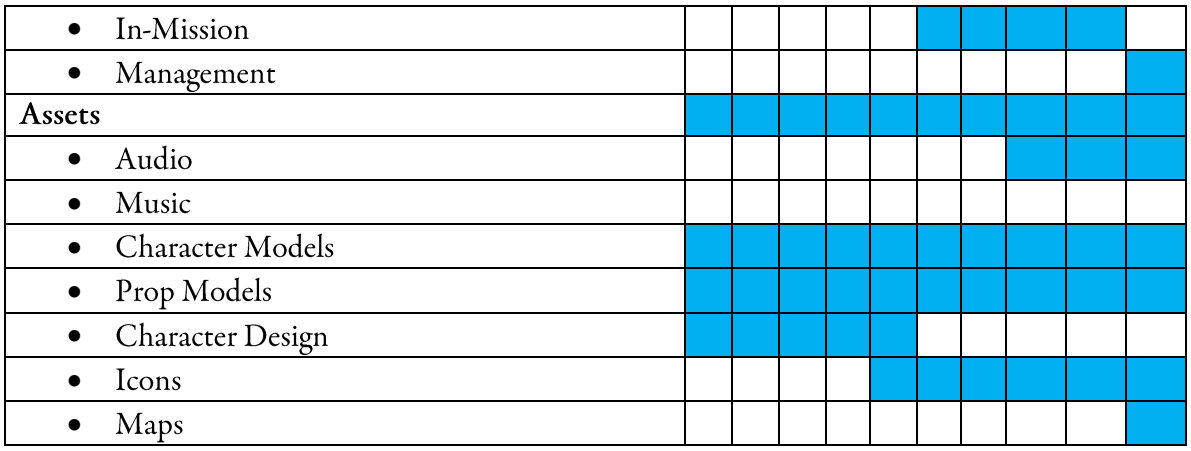


Figure 2. Team 1's Fall Semester Gantt Chart

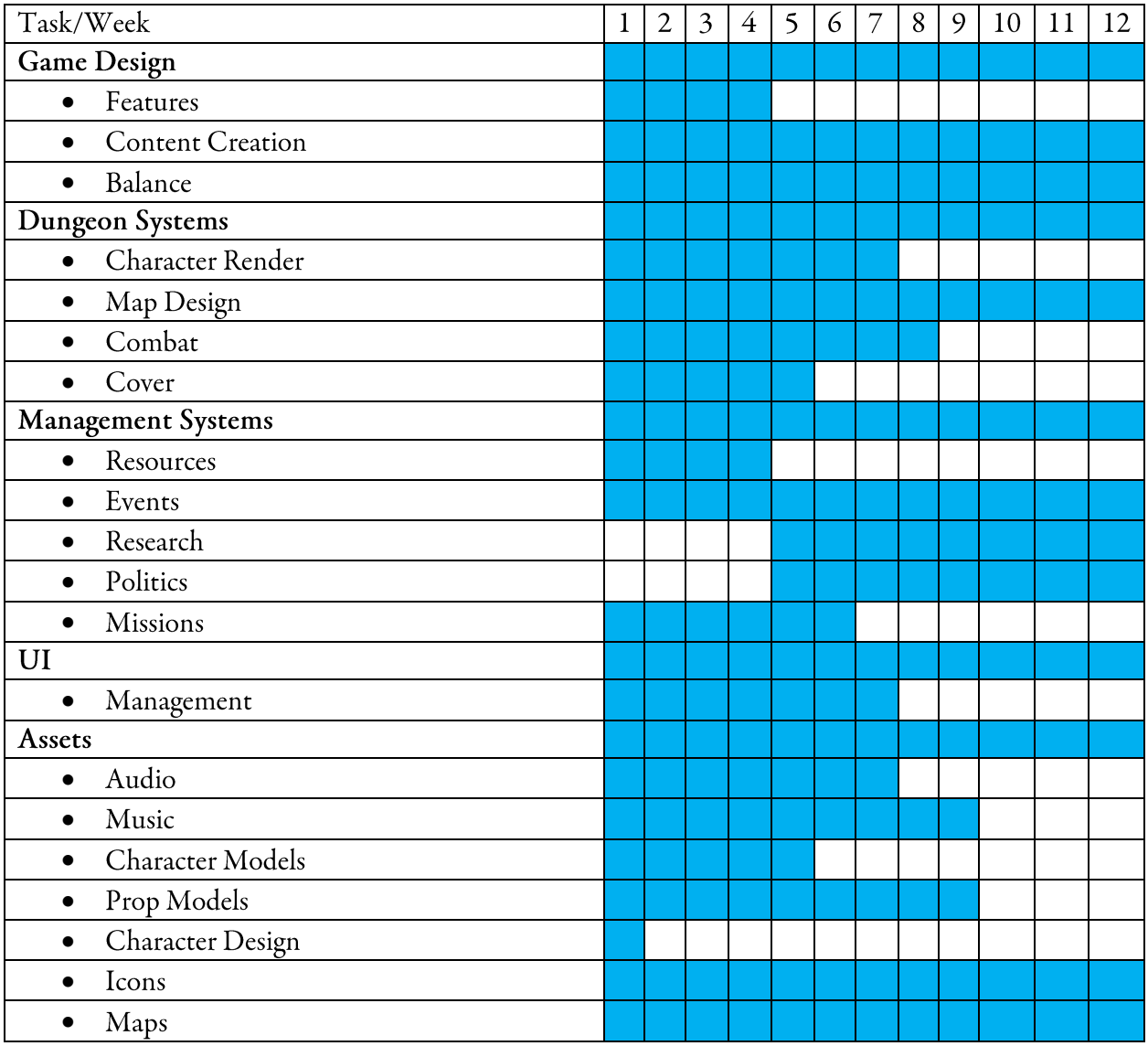


Figure 3. Team 1's Spring Semester Gantt Chart

CONCLUSION

LESSON LEARNED

Since we started this project, we've had upward of half a dozen release versions of Unity3D come out. This specifically left us in desperate need for a reliable source of information the majority of the time, since the official documentations was always one and a half major version behind, and no tutorial or guide found on the internet at large was guaranteed to be working, or continue to work. This ended up with us doing a lot more trial and error processes than we would've liked; sometimes the entire team dedicating time to fix a single problem which was a blocker in someone's weekly report. That said, Unity3D and its popularity also gave us a wealth of resources and pre-made assets to benefit from, as well as the possibilities bundled with it.

We've also personally learned a lot regarding team management throughout this experience. We had many discussions as a team on how to proceed with certain decisions, such as whether or not to do map design as scene-by-scene transitions, or procedural generation scripts. Assets and contents such as items, arts etc. all went through a team channel before being decided upon, and a lot of personal insights from team members helped with the process where a single person could not. Weekly meetings coupled with deliverable-driven schedule for everyone did a good job at keep their peers up to date.

Game design was also a frequent topic which came up during meetings. Despite knowing that we will have to inevitably tighten our scope, it was a surprise to all of us just how drastic this needed to be done. The amount of content (art, graphics, characters etc.) was cut by a significant amount to accommodate delays due to health and personal issues of team members.

Decisions such as mission types, the lack of custom-built systems that were more than just essential combat are also sometimes made in favor of the least work required due to the lack of manpower on hand. A significant amount of testing was also missed due to the toning down of features and the lack of general comprehensive testing methods.

Testing would be something we'd like to focus on in the future, as it stood most of the testing for the game was done by hand. There was no automatic testing system as we were still in alpha phase for the majority of development time, and were still learning a lot along the way. We would also like to expand the number of testing scenarios available in order to encompass more potential testers as well as possible bugs.

Eight months of outside collaboration with people willing to provide effort for free was also a very important learning experience. It's unreasonable to expect an equivalent of passion and priority from those without obligation to contribute or to adhere to a strict deadline as we did, and thus both quantity and quality of outside help for technologically unrelated tasks such as modelling or art deteriorated as time went on. The problem was further compounded by the fact that the content spreadsheet relied heavily on the ability for us to gather assets to represent said content, and with our focus on the technical aspects, actual implementation of these data slowed down heavily.

NEXT STEPS

We plan to take this product beyond the class, post-graduation and toward a commercial release. The product will remain maintained for however long we have available resources and

labor. Depending on its financial viability, we would also release further updates to the game to iron out mechanics, release more content, and balance existing content.

The ground plan had been to carry the game's development process through motions. Implementing features and doing alpha testing for the first two semesters was what we've been doing. Once this was finished, and open beta release to the public would allow us further feedback to work with our customers, and to pitch ideas for potential publishers or resource creators that may be able to help us with the polish of the game. Steam's Early Access program is also something we considered, the success of which could give us the resource and exposure we need to guarantee a quality product upon release.

Bibliography

Farlex. n.d. "Definition of betaware in the Free Online Encyclopedia." *thefreedictionary.com.* Accessed 11 8, 2016. http://encyclopedia2.thefreedictionary.com/betaware.

GameGrind. 2015. "Inventory System Tutorial in Unity 5." *YouTube.* September 8. Accessed 2016. https://www.youtube.com/playlist?list=PLivfKP2ufIK78r7nzfpIEH89Nlnb__RRG.

Quintans, Desi. 2013. "Game UI By Example: A Crash Course in the Good and the Bad." *Evatotuts+*. 1 22. Accessed 11 8, 2016. https://gamedevelopment.tutsplus.com/tutorials/game-ui-by-example-a-crash-course-in-the-good-and-the-bad--gamedev-3943.