BushBurg: Survival in Africa

Final Report and Software Specifications

for

Amapalo, Inc

&

University of Nebraska, Omaha

December 19, 2015

Matthew Callen

Khangal Erdenetsogt

Table of Contents

[1. Background: Mobile Game Development 1](#_Toc438428805)

[1.1 Amapalo: Motivations and Goals 2](#_Toc438428806)

[1.2 UNO Capstone: Scope and Expectations 3](#_Toc438428807)

[1.2.1 BushBurg’s Complete Picture 3](#_Toc438428808)

[1.2.2 A Manageable Scale 4](#_Toc438428809)

[Figure 1: Traditional Design Pipeline for a Video Game 5](#_Toc438428810)

[Figure 2: Reduced Scope for Capstone Project Feasability 6](#_Toc438428811)

[1.3 Technical Challenges and Ethical Considerations 7](#_Toc438428812)

[2. Functional and Non-functional Requirements 9](#_Toc438428813)

[2.1 Game Description 10](#_Toc438428814)

[2.2 User Characteristics 10](#_Toc438428815)

[2.3 Constraints 11](#_Toc438428816)

[2.4 Requirements 11](#_Toc438428817)

[2.5 Flowchart of processes 12](#_Toc438428818)

[2.6 User scenarios/Use case diagram 13](#_Toc438428819)

[2.7 User requirements 14](#_Toc438428820)

[2.8 Functional requirements 15](#_Toc438428821)

[2.9 User interface requirements 16](#_Toc438428822)

[3. Design and Architecture 17](#_Toc438428823)

[3.1 Gameplay Abstract 17](#_Toc438428824)

[3.1.1 Citizen Attributes and Task Allocation 17](#_Toc438428825)

[3.1.2 Task Types 22](#_Toc438428826)

[3.1.3 Other Game Aspects 24](#_Toc438428827)

[3.1.4 User Interface Elements 26](#_Toc438428828)

[3.1.5 Time Scales 28](#_Toc438428829)

[3.2 Software Architecture 28](#_Toc438428830)

[3.2.1 Choice of Unity Engine 28](#_Toc438428831)

[3.2.2 Unity Architecture 29](#_Toc438428832)

[3.2.3 Prefabs and Instantiation 32](#_Toc438428833)

[3.2.4 Information Hiding in Unity 33](#_Toc438428834)

[3.3 BushBurg Architecture 34](#_Toc438428835)

[3.3.1 3D Objects in a 2D BushBurg 34](#_Toc438428836)

[3.3.2 The Utilities Class 34](#_Toc438428837)

[3.3.3 Overview of Object Interaction in BushBurg 35](#_Toc438428838)

[3.3.4 BushBurg’s Classes 36](#_Toc438428839)

[4. Implementation Issues, Risk, & Balance Concerns 38](#_Toc438428840)

[4.1Implementation Problems 38](#_Toc438428841)

[4.1.1 Canvas-Based UI 38](#_Toc438428842)

[4.1.2 WorkStation Inheritance 39](#_Toc438428843)

[4.1.3 Mental Inertia 39](#_Toc438428844)

[4.2 Balance Problems and Recommendations 40](#_Toc438428845)

[4.2.1 Crops, Attributes, & Fatigue 40](#_Toc438428846)

[4.2.2 Problems with Buffs 41](#_Toc438428847)

[4.2.3 A Note on Testing Procedures 42](#_Toc438428848)

[5. Summary 43](#_Toc438428849)

[5.1 Labor Report 43](#_Toc438428850)

[5.2 Risk Report 44](#_Toc438428851)

[5.3 Schedule of Events 46](#_Toc438428852)

[5.4 Lessons Learned 47](#_Toc438428853)

[6. Conclusion 48](#_Toc438428854)

[Appendix A: Crop Statistics 49](#_Toc438428855)

[Appendix B: Setup 52](#_Toc438428856)

[Appendix C: Team Meetings 53](#_Toc438428857)

# 1. Background: Mobile Game Development

Just a few years ago, the mobile games market was a great place for amateur developers to practice their skills and try their luck. Without much curation to speak of, success in mobile games was unpredictable. Initial success often snowballed into a global hit. Many games, if not most however, struggled to make any money at all. This is because of the way visibility within the store was structured. Without something to start people buying your product, it would soon be impossible to find a new game without specifically searching for it.

Game ‘clones’ became a manifestation of this problem. Many well-developed games simply never got very much exposure, and other companies took advantage of this by essentially copying every aspect of the game and re-releasing it under a new name. Their chance was just as good to actually be successful, without any of the effort of making a good game.

As the market developed, companies found that it was much easier to get people to try a game that was free. Instead of charging up front, companies built their games around the idea of small purchases introduced later. These small purchases would enhance the user’s power within the game, allow for more playtime, or save time and provide convenience. These kinds of games were immediately more successful on mobile platforms, and now just about any game that aims for success on these platforms will be free.

In 2015, the top two grossing games on mobile platforms (iPhone, Android, iPad) were Game of War: Fire Age and Clash of Clans. Despite being entirely free to download and play, they brought in over one million dollars a day each. Mobile games account for more than 40% of app downloads, and it is projected that the mobile game industry will be worth over four billion dollars in 2017. What was once a fledgling market is now a thriving industry; big players, and in turn, their large marketing divisions, have secured a footing in the mobile games scene.

Mobile gaming is now a well-recognized medium for gaming and gaming companies. With billions on the line, competition will only get stronger as time goes on. Genres have developed within the market, and core expectations for those genres are influencing development. Mobile games now resemble traditional gaming much more. Marketing budgets and expectations of quality of have greatly increased.

## 1.1 Amapalo: Motivations and Goals

It is easier than ever to feel connected to the world around us—at least, parts it. It is common for us to keep up with news in Europe, Asia, and the Middle East. Many of us have regular interaction with those an ocean apart through online communities. However, increased connectivity between inhabitants of the first world has further disconnected us from those of the third world. It is still difficult to make people care about those with which they have shared no contact. Additionally, the image of charity has taken something of a beating in recent years. Between fraudulent enterprises, counterproductive results, and the opacity of most charity expenditures, it has become difficult for charities to justify their existence to new generations. The pursuit of funding and volunteers for humanitarian efforts must evolve with the 21st century.

BushBurg is an attempt at exactly that type of evolution. Gaming is regarded as one of the most subversive and effective means to understand how it feels to be in another person’s shoes. The medium has also been shown to convince people to invest money willingly on a regular and long-term basis. These two facets of the industry lend themselves very well to raising money for charity. The trick, of course, is to make a compelling game experience which facilitates donation.

BushBurg is the brainchild of Amapalo, an upstart mobile gaming company focused on a new genre of mobile gaming – charity gaming. The founders have built an affinity over the years for the culture and people of rural Zambia, through mission trips with their church. Mobile gaming is a way for Amapalo to tell their story to the world. They have plenty of experience with the region, the people, and with software marketing. The missing piece of the puzzle, for now, is experience with game design and development. That’s where this project comes in.

## 1.2 UNO Capstone: Scope and Expectations

It is tempting to make unattainable goals in game development. The sheer labor involved with even simple games can be quite deceiving, and each new step is harder than the last. The experience of the team in earlier game development projects has colored our understanding of what exactly we can do with the time available to us. This section describes what the client wants, what we can actually do, and what a new team might do with the foundations in place.

### 1.2.1 BushBurg’s Complete Picture

BushBurg is, in its conception, a survival and management simulator. Players are in charge of a growing community of simulated citizens, each of which is imparted strengths, weaknesses, and preferences. Players guide their citizens through the cultivation of food, the building of infrastructure and homes, the training and specializations of citizens, and the tribulations of life in rural Africa. Through smart decision-making and help from others, each player’s community will develop and modernize over the course of months or years.

The game was initially described as a cross between FarmVille and Oregon Trail. While discussions about possible alternatives have moved the overall design such that this description is no longer accurate, some aspects of those two games remain. Cultivation and farming are still a core part of the experience, and so is the risk management of Oregon Trail. Citizens are subject to constant risks in the African savannah, and the management of unfortunate events is an integral part of the experience. However, the game is not about struggle. There is no expectation of the struggles of the player to be true to life, but the idea is to develop a kinship with the simulated inhabitants.

Mobile games are inherently social experiences, compared to more traditional platforms. Many games provide benefits to teaming up with friends or even strangers. BushBurg takes the social aspect a step further. Players, organized in communities of their choosing, are able to trade and donate materials to each other. This includes the game’s monetization structure, which allows players to help their community, themselves, and real people in need at the same time. An in-game purchase might be made, for example, for a community mosquito net. This might have the effect of reducing the risk of Malaria in a specific space. Additionally, if the community purchases enough of them, a real mosquito net could be donated directly to those who would benefit.

### 1.2.2 A Manageable Scale

The goals of Amapalo in BushBurg, noble as they are, cannot be accomplished in a semester. Even a team of professionals, working full time, would struggle to make significant progress. A team of two people with full schedules cannot hope to build a well-rounded game experience in 13 weeks. In addition, many aspects of the process require skill sets that are not developed in Computer Science curriculum. Most of the problematic requirements are involved with the development of art assets. Things like sprites, models, particle effects, sound effects, and music are far outside the expertise of the team. Fortunately, these aspects can be developed at a different time, with a different team. Integration of graphical and sound assets is actually made quite easy by current-generation software. Amapalo has expressed that these aspects are planned for at a later period, and will not require our involvement.

Social angles to gameplay also cannot be reasonably implemented. This would require use of both a server infrastructure and a test community- neither of which is currently available. Instead, our goal is to produce a single player prototype that examines the most fundamental parts of gameplay. We will develop a set of rules which govern how the user manages his or her citizens in a strategic manner. This is a vertical prototype, meaning it fully explores a single set of ideas as strongly as possible. From there, we can fully understand the implications of that set of mechanical rules and how they might be altered to accommodate more complicated rules in future iterations.

We have created a core systems prototype, which is a vertical dive into a single set of mechanical systems. Testing and evaluation followed, with some time left for a couple of iterations on the existing systems. We ignored surface elements, such as graphics and sound, almost entirely. Any graphics or effects used were made for the sole purpose of helping us test the interactions. Future teams may build on these ideas or, if the client wishes, start fresh with a better perspective. The figures below illustrate how the scale of the project was pared down to a reasonable scope.

### Figure 1: Traditional Design Pipeline for a Video Game

### Figure 2: Reduced Scope for Capstone Project Feasability

## 1.3 Technical Challenges and Ethical Considerations

The most difficult technical aspects of developing a game involve the most basic and fundamental procedures which make them possible. Organizing objects and rendering them onscreen *efficiently* is very difficult to do, and would take more time than we had to devote to the project. Thankfully, there is no need to create a custom engine for BushBurg. Very early in development, we chose to use the Unity engine for our prototype. That choice will be discussed in more detail later in the report. Much of the low-level, engineering-oriented code does not vary greatly from project to project, and can be reused. By using Unity, we are able to actually commit our time to the game ideas instead of spending the entire semester reinventing the wheel.

Of course, this does not mean that our job is easy. Game development is a labor-intensive exploration into new ideas. Each new step is more difficult than the last, and missteps in the early stages of development can cripple progress later on. The largest areas of concern, in retrospect, are as follows:

* In order to maintain flexibility early on, a three-dimensional scene was used in development. This complicated the positioning of objects within the world and required accommodations for how things could be selected.
* We were developing a game that, upon completion, would take months or years to experience. Designing and testing systems for that time scale proved challenging, as real-time play is much more difficult to evaluate. Many small changes had to be made to facilitate playability in real time.
* We were limited to ideas we could test in a vacuum. Due to the time constraints, many fundamental aspects of BushBurg could not be developed. Making considerations for those aspects within design, without being able to test them, was difficult.

Due to the fundamentally incomplete nature of a systems prototype, monetization strategy was not considered for this project. However, experience with the market has given us some understanding of the ethical hazards of mobile games. With the free-to-play model, many companies have begun to employ psychological research into their monetization strategy. Humans are rational, but may be susceptible to manipulation when their rational facilities are impaired or ignored. Leisure is a period of time in which humans often intentionally let down their guard.

A comparison can be made to the realm of gambling, where decades of research have gone into keeping users addicted to the thrill of winning, even when they know the odds are stacked against them. Psychological ‘Skinner Box’ effects are often employed in gambling. These tactics space out rewards in a way that makes losing feel like winning. Mobile games have begun to use Skinner Box effects as well in their free-to-play games.

The big difference, and the resulting ethical dilemma, is that mobile games are not age restricted in the way that gambling is. The state of brain development in younger children precludes the ability to moderate Skinner Box behavior, and a game which preys upon this weakness is precisely the opposite of charity. It is important, when the time comes to choose how money will be spent within BushBurg, to specifically avoid these tactics. It is surely possible for a well-designed free experience to be sustainable without them.

In future sections, specific requirements and implementation details will draw a more complete picture of our systems prototype. With skillful design, and some luck, we hope that future teams will be able to bring BushBurg to the exciting new world of mobile gaming.

# 2. Functional and Non-functional Requirements

This section describes the user requirement and functional specifications of the “Bushburg: Survival in Africa” project. The specifications are derived from the user needs research carried out during meetings with the client and requirements document (see figure 1) submitted by them. Usability for the project’s target group of the general public is a key element. To describe the procedures involved, use case diagram is included along with flowchart-type notation. It is the intention here to define functional and non-functional requirements that describe all possible requirements in the project.

While the functional specifications themselves are essentially technology independent, technology choices are required for the development work to follow in building working prototype. Therefore, team skills have been assessed for selection of tools and platform for development.

While the implementation of a game community is out of scope for the project, the prototype must be conducive to community interaction and cooperation. This could be through trading systems, a sub-community system, and cooperative donation strategies.

The game should lend itself toward active involvement in the donation process, through which the player will be made aware of their influence on real world events. For the prototype, this means that in-game donation results should have real world analogs.

## 2.1 Game Description

The working prototype of the game is divided into 9 different levels in terms of processes and procedures which they involve: a board simulating the economics of small crop field. The game’s objective is to raise money by planting, harvesting crops and selling stocks including yam, corn and collard green etc. This is done by selecting stocks, citizen farmer and assigning him to crop fields for harvesting. Players can increase their chances of earning more money by planting more crops. With the earned money, players can purchase more stocks, houses of livestock necessary for his farm. The elements of the game are intended to reflect aspects of community in rural Zambia. The cultivation of land, the perils of the environment, and the hope of investment paying off should all be represented within the game. The list of requirements for stocks, buildings and other necessary items are described in figure 4.

## 2.2 User Characteristics

The target user for the game will be of ages starting from 12. They will be novice game players who understand board games and how to use a mobile device. The target users come from diverse cultural and professional backgrounds.

# 

## 2.3 Constraints

The following are the constraints of the project:

* Touch screen interface constraint: Interaction with the game will typically be done on a phone or tablet device. This means that screen real-estate will be at a premium, and that precision input cannot be expected. The ‘swipe and drag’ functionality of touch-screens does provide a few additional design options.
* Low session time: Whether it can be connected to battery-life concerns or the inherent ‘short break’ nature of phone interaction, we must be careful not to require playtime sessions in excess of 3-5 minutes at a time.
* Criticality of schedule: The project duration is very critical for our team. The prototype needs to be developed within 12 weeks.
* Team size is another big constraint for our team. There are only 2 members in the team.
* Number of players: It is a single player game.
* No graphic and sounds

## 2.4 Requirements

This part contains the user requirement and functional specifications for the prototype. They provide the basis for the design and building of the working-models of the prototype. The levels and associated functions proposed are broken down into 9 levels. The functional specifications are designed to cover all of the functions required in all levels. The processes and procedures which are involved in implementing the functions for each level are described in the form of simple flowcharts.

### 2.5 Flowchart of processes

The diagram used is of the flowchart type. The processes are graphically represented by shapes. The sequence of events is indicated by lines and arrows between the symbols.

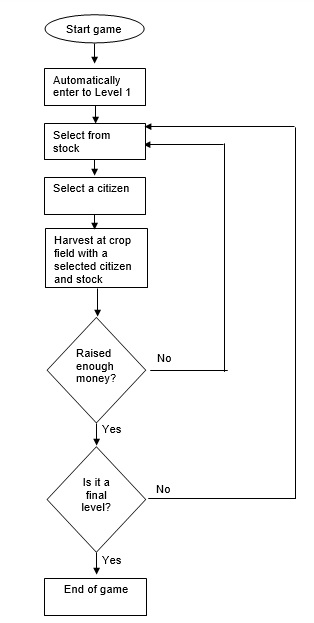


Figure 3: Flowchart of processes

### 2.6 User scenarios/Use case diagram

A use case diagram describes a sequence of actions performed by a player. Here:

* Player: A player is a person that plays a game.
* Associations: Associations between players and use cases are indicated in use case diagrams by dashed lines.
* System boundary boxes: A system boundary box indicates a scope of the project. Here, making purchase during a game is out of scope.

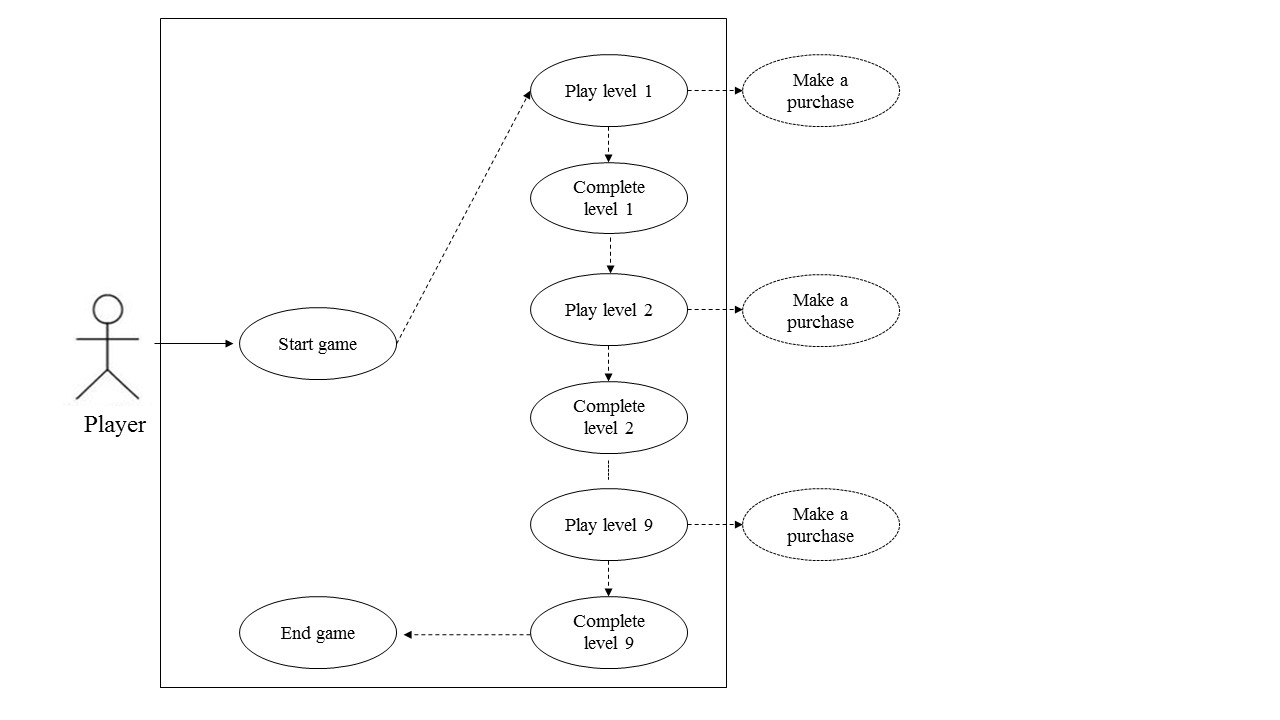


Figure 4: Use case diagram of the prototype

## 2.7 User requirements

In the general approach to determining and describing the user requirement and functional specifications, the client has provided a good requirement document. The meetings with the client were also a great opportunity to define the requirements.

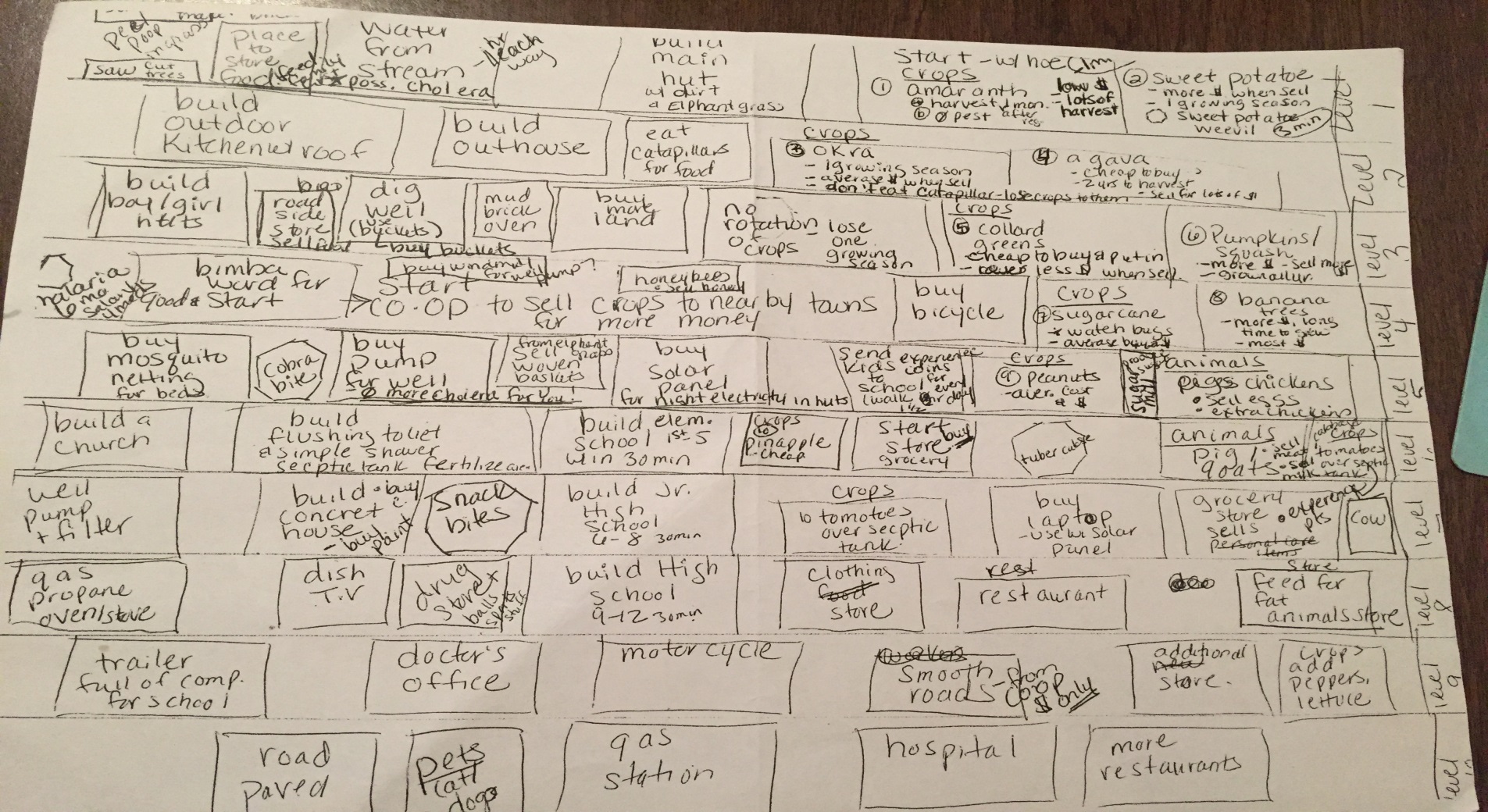


Figure 5: User requirements

The following is a list of stock requirements provided by the client.

* Level 1: Sweet potato, Agave
* Level 2: Okra, Pumpkins
* Level 3: Collard greens, Squash, Chickens/Eggs
* Level 4: Sugarcane – water bugs, Banana, Bees/honey
* Level 5: Cabbage, Peanuts, Pigs
* Level 6: Pineapple, Goats/milk
* Level 7: Tomatoes, Cows
* Level 8: Lettuce, Peppers

## 2.8 Functional requirements

The mobile marketplace for games is volatile and competitive, with failure often being absolute and success rare. Care must be taken to consider the nature of the market and what has been shown to work within it. The needs of the market and the needs of the client may not be inherently compatible, so compromises are made. The following is a list of requirements imposed by the medium, the market, or the client, in that order.

* *Touch-Screen Interface (constraint of medium):* Interaction with the game will typically be done on a phone or tablet device. This means that screen real-estate will be at a premium, and that precision input cannot be expected. The ‘swipe and drag’ functionality of touch-screens does provide a few additional design options.
* *Low session time requirements (constraint of medium/market):* Whether it can be connected to battery-life concerns or the inherent ‘short break’ nature of phone interaction, we must be careful not to require playtime sessions in excess of 3-5 minutes at a time.
* *Free access to play (requirement of market):* Even well-reviewed titles with costs of entry almost universally perform poorly on mobile markets. This means that the monetization model of the game will be what is dubbed “freemium,” with small transactions offered to enhance the player’s experience throughout. This has extreme implications for the mechanical design of the game, and will be probably the largest design challenge.
* *Motivation for long-term investment (requirement of market):* Free-to-play titles get much more return from players that have been involved with the game for long periods, often in excess three months. The game must be designed such that it is continuous and cannot be completed quickly.
* *Slow increase in complexity (requirement of market):* The game should not overwhelm new players with complicated systems. Since there was no initial cost, players which are unable to understand the game immediately will simply uninstall it. However, given the previous long-term investment requirement, the complexity does need to exist at some point to keep players interested in improvement. This simply requires a careful approach to the introduction of new complexity.
* *Perception of fairness (requirement of market):* All items or currencies available for purchase for real money must be otherwise attainable. If this is not the case, the game will likely be written off as predatory by users. Generally this balance manifests in terms of time investment vs money investment.
* *Bias toward success (requirement of client):* The game, while respecting the challenges of daily life in rural Africa, is expected to emphasize the positivity and hope of its residents. As such, it should not be punishingly difficult, and will have a ‘little victory/little failure’ ratio of roughly 80/20. Management of this ratio can be difficult for a simulation, and will require testing.
* *Community focus (requirement of client, planned but not implemented):* While the implementation of a game community is out of scope for the project, the prototype must be conducive to community interaction and cooperation. This could be through trading systems, a sub-community system, and cooperative donation strategies.
* *Real-world tracking of donations (requirement of client, planned but not implemented):* The game should lend itself toward active involvement in the donation process, through which the player will be made aware of their influence on real world events. For the prototype, this means that in-game donation results should have real world analogs.

## 2.9 User interface requirements

The user interface is built around the following aspects:

* Main screen: where player checks stocks, citizens, crop fields and money raised.
* Crop field: Here player allocates tasks to raise money
* Stock: A list of available stocks
* Citizen: A list of workers

# 3. Design and Architecture

The following section concerns the design of the BushBurg prototype, both in abstract gameplay terms and the implementation within Unity. Gameplay is described first, providing an understanding of the goals of implementation and a foundation for future iterations of design. Next, the Unity engine is discussed in a broad sense, with rationale given for choosing its use in this project. Finally, the implementation of the mechanical systems will be shown, as a primer for any developer that may wish to continue with these systems.

## 3.1 Gameplay Abstract

Management games tend to attract players that enjoy efficient progress. To put it another way, they like to be offered many small choices; choices that add up over time to a large difference between doing well and doing badly. The core system implemented in this prototype is the primary engine for these little decisions over time. Below are some guiding principles that lead us to our choice.

* A player that plays more should progress faster, but one that is less active should still progress
* Complexity should ramp up slowly over time
* The barrier to entry should be low in terms of required skill
* There should be some sense of individuality for each simulated citizen

### 3.1.1 Citizen Attributes and Task Allocation

Each citizen may be assigned a task at any given time. The player will want to assign the right task to the right person, in order for things to work as efficiently as possible. In order to make that decision a gameplay element, there need to be rules that suggest which task is best for which citizen. To this end, we have developed a set of attributes that governs how well certain tasks are performed. For the purposes of this prototype, the attributes will be fixed. This will allow us to test the systems affected by the attributes more closely.

Efficiency Attributes

Each citizen has three attributes associated with efficiency. Higher levels of the attribute mean that the task will be performed at a faster rate. These attributes are strength, dexterity, and endurance, which are fairly common attributes in games.

Quality Attributes

Each citizen has three attributes associated with quality. Higher levels of the attribute mean that the outcome of the task will be better. The implications for quality depend on the task involved, and are not necessarily important for every task. There should be some cases in which quality is more important, and some in which efficiency is more important. The quality attributes are perception, focus, and acumen.

Task Attributes

Each task will involve up to all six attributes in some way. In respect to a given task:

* One primary (draining) attribute - Primary attributes for a given task will affect efficiency and quality the most. While the task is being performed by a citizen, that attribute will slowly drain, making them worse over time. This works as a percentage of the citizen’s maximum.
* One secondary (locked) attribute – Secondary attributes will affect the task less than primary attributes, and cannot be recovered while assigned.
* One unused (recovering) attribute – The third attribute does not affect the task at hand at all, but will recover over time, up to the citizens natural maximum.

The purpose behind this system is to create a way for players to optimize the village through micromanagement. With no other influence, the best way to play would be to rotate tasks by attribute, in order to prevent the loss of efficiency through fatigue. Idle citizens would recover all three attributes, so there could be some utility in allowing idle time as well.

Global Attributes

Three additional attributes exist which influence the task-centric attributes. The management of these is more gradual and overarching. These are:

* Health – A representation of the overall health of the citizen. This would usually change due to things like disease, poor sanitation, poor water quality, etc. The health attribute affects how quickly all efficiency attributes drain. A low health citizen will lose strength much faster than a high health citizen when performing a strength oriented task.
* Happiness – This functions very similarly to health, but instead affects quality attributes. Like health, it is inversely correlated with quality attribute drain.
* Recovery – This represents the citizen’s level of hunger (it could also be called satiation). This attribute affects the rate of recovery of all task attributes. A well-fed citizen will recover from fatigue much more quickly.

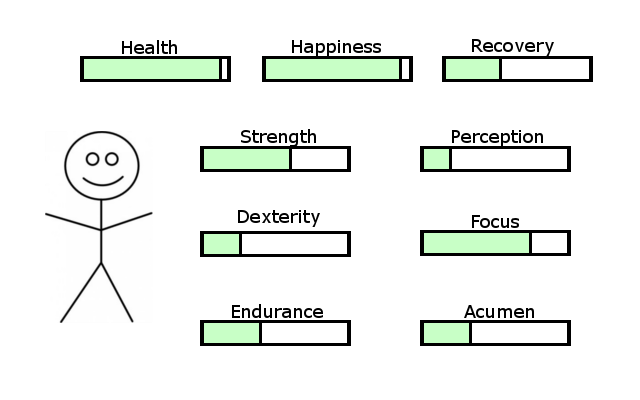
Food and its Effects on Attributes

One of the major intended features of the game is nutrition and cooking. The many crops that are available as the village progresses could be used in recipes. Recipes and their assembled meals would afford specific, timed benefits that make certain things go more smoothly in the village. For this prototype, recipes were not going to be implemented. Instead, the benefits of the food are more direct. A given food, if eaten raw, will simply increase the recovery attribute by an amount. This is useful for recovering from fatigue, but the real goal of the food system is to use cooking to improve food quality. A cooked food item will also generate one of the timed benefits mentioned earlier. These timed benefits will be unique to each food type and will be affected by the quality attributes of both the chef and the cultivator.

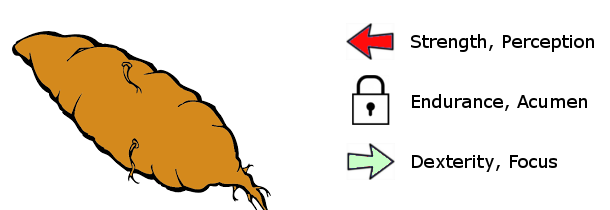
**An Example Scenario**

For the moment, consider only cultivation, cooking, eating, and idling as possible actions for citizens to pursue. Below is an example citizen, followed by a possible set of tasks and food interaction.

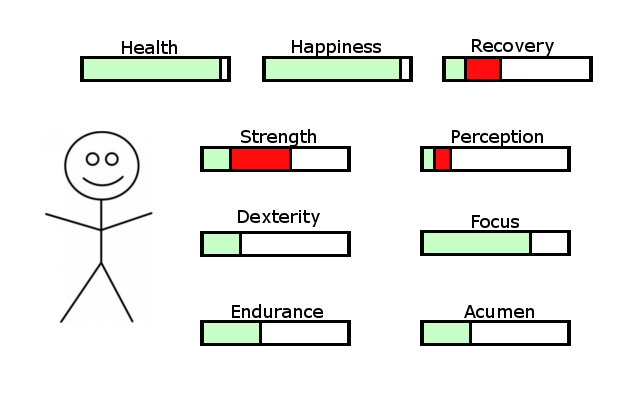
This citizen is healthy, happy and strong. He would be best used in a task that requires strength and focus as primary attributes.



In this case, we don’t have a good option for both, but we can cultivate sweet potato quickly with our high strength attribute. Our citizen needs to eat soon. The primary attributes for sweet potato cultivation are strength and perception, which will drain as the farm plot is worked.



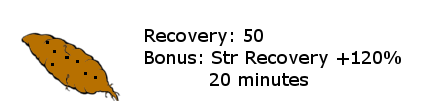
After some time, our citizen’s attributes have been modified by working the sweet potato plot. Our citizen is even hungrier now as well.

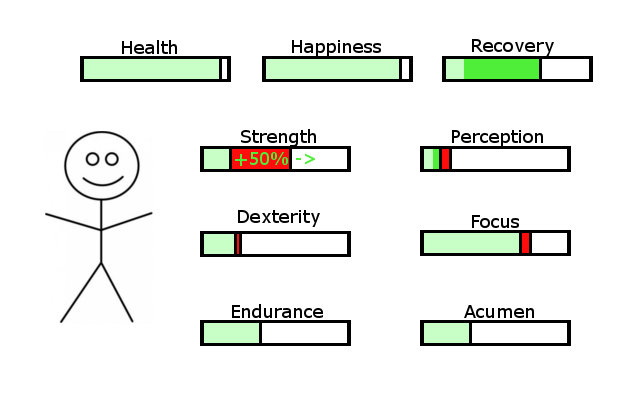


We could eat the potato and gain recovery, but there is a better solution available. Cooking the potato will produce a buff effect which increases strength recovery by 50% for 20 minutes. This task also benefits highly from focus, which will improve the quality of the food.



The cooking was a success, though at some cost to our citizen’s dexterity and focus. Focus and acumen have increased the quality of the result as well!





The player would most benefit from letting this citizen rest idle a bit, or assigning him a task that does not use strength. If assigned a task which drains or locks strength, the special bonus of the cooked sweet potato will not be utilized.

### 3.1.2 Task Types

Growing and cooking crops comprise most of the gameplay and strategy of this prototype, but other tasks do exist. Further development should aim to introduce many more tasks. All existing tasks, as well as some of those intended for future development, are listed below.

* Idle/resting: In this prototype, there is a designated area for intentionally idle workers. In the long run, ‘idle’ would refer to anyone that is not actively participating in a task that drains attributes. In the idle state, recovery is boosted. Currently, this is set to 2x recovery. In residences, this is boosted even more.
* Cultivation: The primary task for this prototype. Fields may be worked by one or two workers, each of which contributes to the pool of attributes used by the field. Pests exist within fields, which will be discussed at a later point.
* Cooking: Citizens can be assigned to cook stored crops one at a time into meals. Meals provide temporary buffs, and are affected by both the quality of the base ingredient (duration) and the quality of the cooking process (effect).
* Trading: This is our means of creating a win condition for the prototype. Crops and meals can and should be sold. Quality of the items is not important; only the quality attributes of the trader are considered. In future builds, trading would be completely reworked.
* Eating: Designated tables exist in order to feed citizens, and they are assigned both a meal and a citizen in order to function. Both raw crops and cooked meals can be assigned to tables.
* Forager: Pests and caterpillars, both explained in detail later on, must be perceived by citizens randomly. The forager will patrol between farm plots and has a greatly increased chance of revealing them.
* Materials Recovery: Construction requires materials such as mud, elephant grass, and lumber. Citizens assigned to this task may be set to gather the specified ingredients.
* Construction: Currently, this task only exists to build houses from materials. Later builds will expand this task to include workstations, etc.
* Training(not implemented): Citizens should be able to improve their attributes through various types of training. This would incur a large time cost, and perhaps some currency as well.
* Water management (not implemented): The gathering and management of clean water is a big part of life in rural Zambia, but we did not have time to address it in this prototype. Water management should evolve over time and take sanitation into account. This would have an effect on the ‘health’ attribute.
* Nurse (not implemented): Hazards, which are only partially implemented in this prototype, should have many lasting effects and need to be managed in different ways. Nurses would aid in the recovery from hazards like snake bites or broken limbs, as well as keep the health of citizens higher.
* Hunting (not implemented): The gameplay of hunting was not developed by this team, but may be beneficial to the game.
* Scheduler (not implemented): When development of a village reaches a certain state, a task may be necessary to reduce micromanagement by the player. If too much micromanagement is required, users may find it tedious to keep up. A scheduler might be able to automate things like eating or resting.

### 3.1.3 Other Game Aspects

Gameplay could be greatly enhanced and further developed in numerous ways. Some possible considerations are discussed later in the report. This section includes mechanics that do exist within the current prototype by some measure.

AutoGather

When a task such as cultivation, trading, or cooking is complete, a gatherable crop is created which can be clicked/tapped to collect it. The collectable item will remain at that position for up to 1/3 of the production time before it is automatically processed. The significance of this mechanic is that it benefits users that are currently online and active. The tasks will complete eventually, but incur a penalty for passivity. This is one area in which some sort of microtransaction might be utilized.

Color Modes

When citizens are assigned to a task, they are given a ‘fitness’ level for that task. In order for us to keep up with the fast pace of the prototype, we included two modes which represent the fitness of the citizen for a given task visually. The color of the citizen will change to some mix of red, yellow, and green. Green citizens are most appropriate for the task- their primary and secondary attributes match the task very well. Red citizens are not well-suited for the role, and yellow citizens are somewhere in the middle. Two color modes exist to help keep track of fitness.

CurrentFit mode will assign fitness values to the task in which the citizen is currently assigned. This is the default mode, and is a good way of keeping track of fatigue. As citizens get tired, they will change color from green to red. BestFit mode utilizes the selection of a specific task, coloring all citizens in accordance to their fitness to that task. This mode is very good for finding the best citizen for a given job

Caterpillars

One of the big pieces of local character associated with the region is the consumption of caterpillars. A feature was requested by Amapalo to represent this in game with a unique mechanic. At any given point, 10 caterpillars exist within the game space, wandering randomly from place to place. Each citizen has a chance to discover them at any moment, based on their perception attribute. If they succeed in the perception check, the caterpillar will be revealed for 10 seconds. During this time, they may be collected by the user and fed directly to citizens, increasing recovery. This does not require use of a table, and is a good way to keep people topped off while working.

Pests

Fields contain pests at all times. These pests are hidden, and have a chance to be revealed upon the completion of a crop. Initially, pests do not have any harmful effects. However, as crops are harvested, the pests will slowly begin to damage the quality of the crop that is harvested. When revealed, the amount of damage done to the quality is shown onscreen. The player is able to redirect the workers of the field to eradicate the pest, which takes some time. Foragers, who patrol the fields, will also automatically eradicate pests that they notice. Crop rotation reduces the effect of pests as well.

Buffs

Four buff types currently exist within the game, each of which affects a single attribute in some way. In future iterations, it may be wise to differentiate buffs and make them more unique. This would be easier with a working recipe system. The four buffs and their effects are as follows:

* Recovery buffs: These buffs increase the recovery of their given stat by percentage.
* Drain buffs: These buffs reduce how fast an attribute is drained by assigned tasks.
* Scalar buffs: Scalar buffs increase the maximum of an attribute by a scalar value.
* Lock buffs: These buffs will prevent a specific attribute from going lower than X.

Hazards

Due to time constraints, hazards are only partially implemented. The intention of this mechanic is to introduce random penalties which can be affected by the status of the village. Some hazards include Malaria, snake bites, broken limbs, etc. The risk of these events may depend on assigned tasks. Broken limbs, for example, are most common in the materials/construction tasks. Snakes are most often encountered in farm plots, trade runs, and water gathering tasks. There are many other possibilities.

Leveling Up

In order to ‘win’ the prototype, users must level up by collecting a certain amount of money from trades. This is unique to the prototype and is not intended for future builds. It exists simply to create a win condition for testing. Each level adds a citizen and a field, as well as unlocking some new crops.

### 3.1.4 User Interface Elements

Interface panels are currently fixed objects within the game world. Future iterations would need to make considerable changes to how the panels are arranged. For the most part, these UI panels exist to help the developers test more than anything else.

Storage Panel

This is the only panel that features interactive elements. A list of all possible crops that can be planted is displayed, increasing in size as new levels are obtained. Players drag a given plant from the storage to the field to plant a given crop. When a crop is harvested, the storage value for that crop is increased, indicating that the crop exists and is being stored. Plant and animal goods can then be dragged from storage to tables, cauldrons, and depots. The quality of crops is currently automatically ordered. Cauldrons and tables receive the highest quality crops available, while depots receive the lowest possible. This panel is too large for constant visibility, and we suggest changing its function in future builds. Planting in fields could more easily be represented by a radial menu which shows up when a specific place is touched. The overall storage screen might exist, but as a pop-out menu instead of constantly being in view.

Current Citizen Panel

In the bottom left corner is a display of the current selected citizen. A segmented bar for each attribute is given. The number of segments is based on the maximum value of the citizen. Some citizens have higher maximums, which differentiate them for various tasks. Red segments will exist when the current value for the attribute is lower than the maximum. Attributes are stored as floats by the game code, but use the ceiling for task calculations. Therefore, a citizen with 8.01 strength, for example, would be considered as having 9 strength for tasks.

Arrows are also displayed on the current citizen panel to represent what is happening to their efficiency/quality attributes. Recovering attributes are represented by a green arrow, while draining attributes have a red arrow. Locked attributes are represented by a lock icon.

Current Task Panel

Players are able to select both a citizen and a task at the same time. The selected task will display information about which attributes it uses, as well as the current efficiency and quality statistics based on the citizens assigned to it. Additionally, the time needed until the next production is complete is shown on this panel as well.

Current Buff Panel

This panel is controlled by the selected citizen, and will show relevant information about buffs. This information includes the duration, the effect, and the type of effect in question. Buffs also have a particle effect which can be interpreted without selecting the citizen. However, only the duration of the effect is obvious from the particle effect, so this panel helps to understand how buffs are being processed.

### 3.1.5 Time Scales

The numbers which govern the speed of the game and how various interactions scale in respect to each other are located in the Utilities script. These are static elements that may be changed to speed up or slow down certain aspects of play, such as the rate of fatigue or certain bonus features.

## 3.2 Software Architecture

This section details the Unity engine and its role in organizing game objects. It should not be considered a full primer on the Unity engine, but will clarify how was utilized for this prototype. Rationale for decisions made along the way will also be featured here.

### 3.2.1 Choice of Unity Engine

Because the development of a game engine takes months of full time work by many developers, it was entirely unfeasible to create one for this prototype. Because of this, we needed to make the decision to adapt an existing engine. The dimensions for consideration are evaluated below.

* Experience: Having previous experience with an engine is an obvious benefit to streamlining development. This is the most important part of the decision.
* Graphical heavy lifting: We could not commit the time to working on basic low-level rendering. Considerable investment is required to get these aspects working in performant way.
* Platform independence: Development and testing is most easily achieved in a PC environment. However, this game will be released on mobile platforms. The engine should be able to build to multiple platforms without much fuss.
* Stable support: Engines are only as useful as they are supported by fresh development. If the engine is no longer supported by the time the release of the game is scheduled, problems with compatibility may come up.
* Free to use for our purposes: We do not have the resources to obtain an expensive license for the sake of a prototype. Additionally, price barriers are of concern to the client.

The dimensions above eliminate many choices by default, leaving only a few choices that are appropriate for our prototype.

GameMaker: This engine is somewhat easy to use, and does the graphical heavy-lifting. However, GameMaker uses its own scripting language and struggles with platform independence.

Unreal: This engine supports most, if not all of the dimensions above, and has gained traction as a very good engine to use. However, it is primarily geared toward first-person perspective, and neither of the team members have any experience with it.

Unity: Unity is free for personal use, very easy to port to new platforms, vary stable in support, and both team members have used it before. It is a natural choice, and has proven to be a good one over time.

### 3.2.2 Unity Architecture

The Unity engine makes use of object-oriented principles to make the development process much more streamlined. Inheritance, polymorphism, overloading, and information hiding all come into play regularly. The environment also contains an editor, which supports a real-time, three-dimensional view of the game scene. Developers are able to directly manipulate objects and cameras and see how the effects will be viewed. Development to testing can be done within the editor immediately, without having to build a binary. Details about the engine are below.

The GameObject

Every new object created within the game inherits from a class called GameObject. This means that certain variables and functions are inherent to anything instantiated within the game. This serves as both a means of improving functionality through libraries, and as a way to employ polymorphism with respect to object interaction. Any developer which uses Unity must become very familiar with the GameObject class. Information detailing this class, among many others, is available at the Unity3D website. Additionally, if using the default Unity IDE, MonoDevelop, users can directly reference a function or field through shift + ‘.

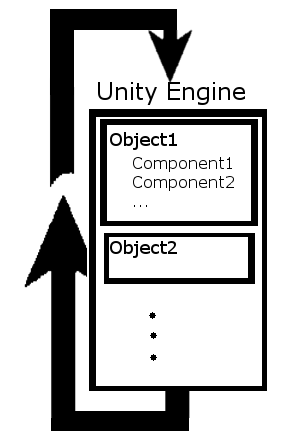
Components

Every game object can make use of various libraries in order to facilitate cohesion within program structure. While it is surely possible to develop custom components instead, there is not often much reason to do so. Default components can be enhanced with extension methods, and contain efficient code that is built to work with the engine on the lowest level. Custom components are likely to be much less efficient. Components are simply classes that can be instantiated within a game object to give it new properties and functions. Some examples of components utilized by our prototype are below.

* Transform: This component contains information about the location, rotation, scale, and velocity of an object in world space. The transform is a core component of any game object, and should be well-understood by anyone that wishes to make a game in Unity. Additionally, the transform is the component that holds references to ‘child’ objects.
* Renderer: This component is responsible for whether an object is drawn on screen, as well as its visual characteristics. Objects without a renderer component will be invisible. Renderers can also be disabled or enabled in order to temporarily hide an object.
* Collider: Every frame, the Unity engine will efficiently test all colliders for any possible overlap. When overlap is detected, a collision object is created with information about the event. Developers can intercept and use the collision objects in order to trigger functions or variable changes.
* RigidBody: While colliders are a separate component, a RigidBody component is also required to give objects a physical presence in the game world. RigidBodies are processed by the physics engine at fixed intervals, every 0.2 seconds. We do not use the physics engine, but require use of RigidBodies for collisions. Checking the ‘is kinematic’ box within the editor on this component will disable physics from being calculated.
* Materials: These contain texture information and control how the ‘skin’ of an object is drawn. Materials must be applied to an object to change how it looks.
* Scripts: Scripts are treated as a component, and are simply classes written by the users of the Unity engine. Unity supports several languages, and we have chosen C# for our scripts because it is most familiar to us. All of our coding belongs in these components, and all interaction logic is written here.

The Unity Game Loop

Running games are treated largely as state machines. First, the status of the game is interpreted, then a block of code runs. All enabled objects within the scene are placed in some order and processed every frame. This occurs as fast as the engine is able to do so. An illustration of the Unity game loop is shown in figure 6.



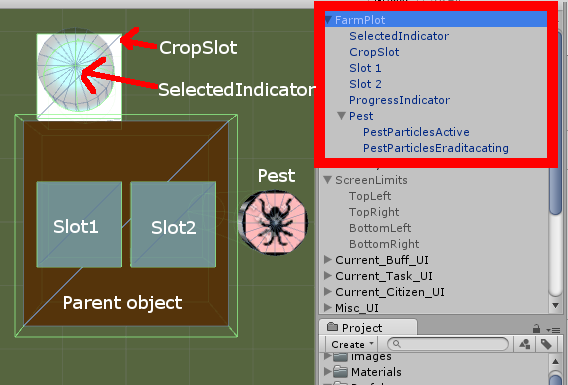
During the game loop, some functions of GameObject are called regardless of specific logic placed there during scripts. These functions are detailed below.

* Update: Code which runs every frame, early on in the frame, should be placed in the update function. This is where input detection is done, since input is event-based and must be polled. Any variable which is updated via a timer would also require processing during the update function. The space between calls to update is not set, and can vary. Utilizing the time.deltaTime field is highly suggested in order to normalize timers to seconds instead of frames.
* FixedUpdate: This is used in a similar way to Update, but on a fixed interval of 0.2 seconds (though this can be changed in the editor). FixedUpdate is primarily used to calculate physics interactions, and is not needed for our prototype.
* LateUpdate: Similar to Update. After all objects have run Update, a new round of updates is done in this function. This function exists in order to avoid calculations from being dependent on the order of Update calls. Renderers get their information from LateUpdate, after all other calculations have been made. Because we don’t use fixed update, and most interactions are sequential per object, this function does not see much use in our prototype. If problems arise in which objects appear to be jittering or delayed by one frame, some updates might need to be moved to LateUpdate.

### 3.2.3 Prefabs and Instantiation

One major advantage to the Unity engine is the designation of prefabs. Prefabs enable grouping of objects into local arrangements of objects with a parent-child reference relationship. The local arrangement, as well as any components or scripts enabled by the objects, can be saved in the file structure and instantiated with code or through the editor. Use of prefabs prevents any need to set up complicated constructors to create new objects. It also enhances inheritance and polymorphism. An example of a prefab is shown in figure 7.

Figure 7: FarmPlot Prefab



### 3.2.4 Information Hiding in Unity

The privacy status of variables and functions within game development is often more open-ended than most software. In Unity, developers may desire to edit values for certain fields within the editor screen. This can be to set references or to tweak values on the fly. In order for values to be changed in the editor, the variables in question must be set to public.

Outside of editor-driven variables, information hiding is highly advised. If all variables are public, the temptation to directly edit variables of other objects within scripts can be toxic as time goes on. Bugs get very difficult to chase. Therefore, all variables should have private setting unless they are editor-driven. Functions, on the other hand, are often public, because interactions between objects are extremely common. One object will need to influence another to avoid constant polling. If an object needs to modify another object’s fields, use a method instead of a public variable.

## 3.3 BushBurg Architecture

This section details the choices we have made to organize objects and interactions within the prototype. While we believe that the reasoning behind the choices is sound, different teams may wish to manipulate how the code is structured. The ways in which objects can be stored and arranged are myriad.

### 3.3.1 3D Objects in a 2D BushBurg

The first architectural decision may seem paradoxical. We chose to use a three-dimensional world for Bushburg, but made use of an orthographic camera. This means that there is no sense of perspective in the camera, and the world appears flat from above. Initially, we were unsure whether a top-down perspective was the desired implementation of the game’s view. To stay flexible and able to change to an angled view if needed, we decided to use 3D objects for our early iterations. Even though we eventually decided to stick with a top-down perspective, the prototype remained 3D.

The means by which selection is done (clicking the mouse or tapping the screen) is influenced by this decision. When a click is made by the user, the location on the screen is projected downward as a ray. One can imagine this as the mouse pointer being a physical object that hovers above the scene, firing downward when a click is made. Rays interact with colliders, and report the first object that the ray collided with. Selection is then made depending on the object which the collider event returns. Ray-based selection would not work in a 2D environment, and thus a different means of selection would need to be implemented if a change to 2D space is made.

### 3.3.2 The Utilities Class

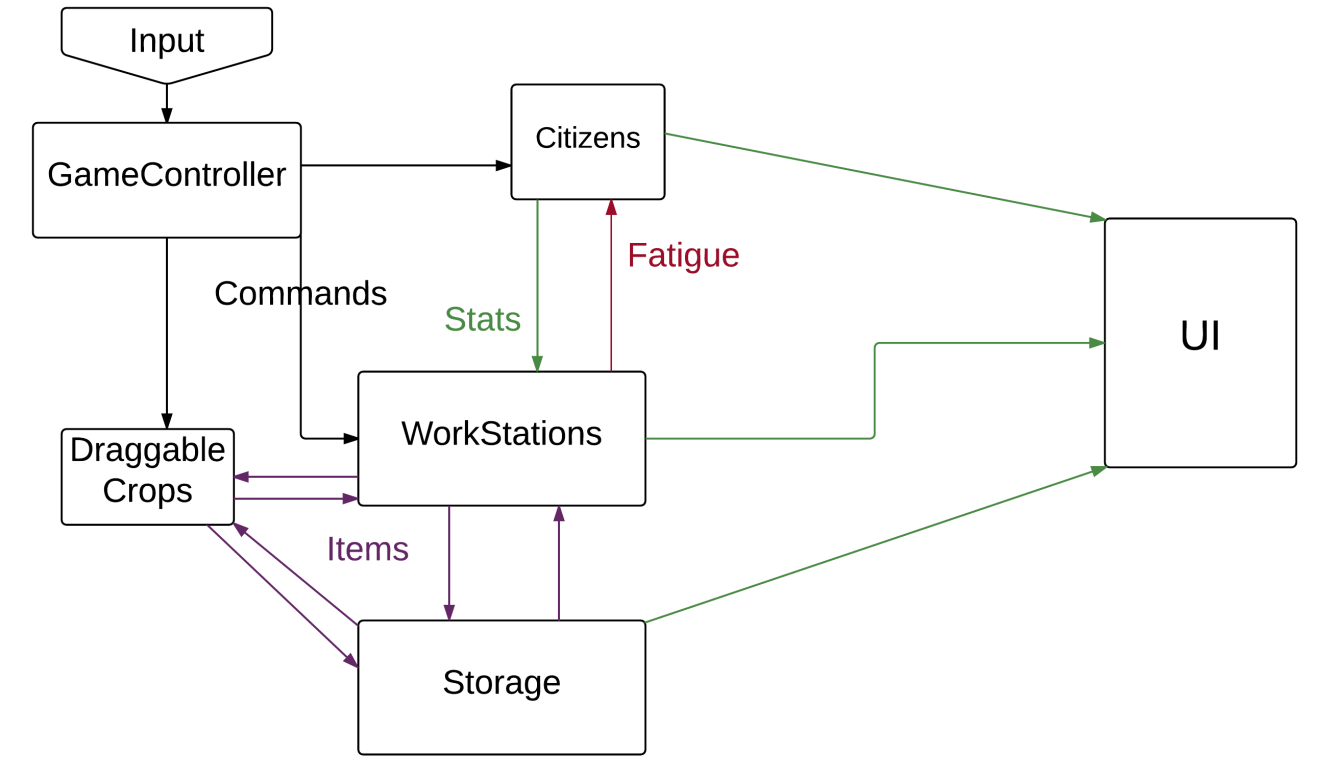
BushBurg takes advantage of static classes within C# by using a static class called Utilities. This class does not require instantiation and does not physically exist in the game. However, variables and functions can be placed within Utilities which can then be called by any object within the game at will. This has the advantage of generating some global constants that all objects abide by. Things such as the timescale, how fast fatigue is applied, and other constant effects are contained in the Utilities class. Additionally, some functions exist in Utilities to speed up common function calls.

The Utilities class also contains many enums. Enums are useful because they provide error-prevention and can be called both by index and by name. There are enums for aspects of play such as the types of crops, the types of work stations, etc. Since IDEs such as MonoDevelop and VisualStudio offer auto-completion during coding, enums are actually faster to use than strings.

### 3.3.3 Overview of Object Interaction in BushBurg

In game programming, runtime efficiency is among the most important concerns. If the game loop is not capable of iterating fast enough, gameplay stutters and the experience of play is unpleasant. This means that the simplest way to code most things in game development is typically unviable. Objects within the game must minimize the use of polling. Polling is iterating through all instantiations of other objects in order to test values every frame. This means that objects must trigger functions in each other regularly, based on events. This can become confusing for the developer to keep track of. Figure 8 contains a visual overview of how objects interact in BushBurg.

Figure 8. Interactions



### 3.3.4 BushBurg’s Classes

A brief discussion of the primary scripts or script types, as well as rationale behind their use, is explained below.

GameController

Use of GameController objects is common in game design. This invisible, unique object in the game typically controls the flow of the game in regard to input and miscellaneous, globally affecting variables. In BushBurg, the GameController handles input, leveling up, and selection. Most other objects contain a reference to the GameController that is set up during initialization. The Utilities class also has a function for obtaining the GameController. Other objects contain this reference because it provides access to selections and to global variables. Other objects do not manipulate these values, but simply use them in calculations.

CitizenBehavior

All of the code associated with citizens is contained in this class. The attributes of the citizens, as well as any modifiers to status, are located here. Functions exist to select, assign, and manipulate attributes. Assignment is handled through the use of collisions. When a citizen is being dragged around by the GameController, it will collide with workstations. When this occurs, a reference to the station is stored as a prospect. When the player lets go, CitizenBehavior attempts to assign the citizen to the station in question. If able, it will call any relevant functions in the workstation, depending on its type. If unable to assign, it will return to its previous assignment. The code used to color citizens based on fitness is also located here.

WorkStationBehavior

This class is a parent class to all types of tasks within the game. Because classes are data types, and C# is strongly typed, the data type is needed before any functions from the script can be called by other classes. If each type of workstation is its own separate class, control statements must be used for any event which is differentiated by types of workstations. This quickly becomes unfeasible as the number of types of workstations increases.

On the other hand, using a single workstation class, control must be employed within the class itself. When a lot of station types exist, the size of the script can easily get out of control. Instead, a polymorphic WorkStation class is used. All of the different scripts inherit from WorkStationBehavior and thus can be declared as such when setting references or calling functions outside of the class.

DraggableCrop

This class performs in a similar way to CitizenBehavior in how assignment works, but instead involves crops and other task items. The creation and destruction of draggable crops depends on the workstation doing the creation or assignment. If a draggable crop is a meal, it will persist for some time in the designated area called MealTray.

StorageBehavior

Storage of crops is handled through a separate entity called the Storage object. This contains information about the number of each item in storage, as well as quality information (stored in an ordered list). The storage object creates Draggable crops when interacted with by GameController. The storage object exists because of the need to store quality information about crops. A rework of the UI may necessitate removal of this class.

Citizen/FarmPlot Managers

A few classes exist with the Manager suffix. The functionality of these classes is somewhat limited, and they are in some sense vestigial. The creation and deployment of farm plots or citizens is initially handled through these classes. Additionally, they populate global lists of all instances of that type object. Some classes, such as the CaterpillarBehavior class, make use of these lists. They could also be located in the GameController class, so future teams may choose to phase these classes out of existence or improve upon their function.

UI Scripts

Each UI panel performs a little bit differently and contains customized scripts. These scripts will draw information from various sources. It is somewhat inevitable that UI scripts will do a lot of polling because they need to update every frame. However, it is possible to streamline them into an inheritance structure. There simply weren’t enough UI elements to require an inheritance structure in this prototype.

# 4. Implementation Issues, Risk, & Balance Concerns

Game development is often quite experimental to all but the most veteran of developers. One never knows exactly what is needed to begin with, and certain structures within the code may become obsolete as time goes on. Design goals may change as well; problems within the logical design of gameplay also may become obsolete. This section details a few of the major problems we ran into during development of this project.

## 4.1Implementation Problems

Many of our initial programming and implementation methods evolved over the course of development. Detailed below is a list of the most pressing issues encountered. Many of these issues were known risks, but some unexpected problems also surfaced.

### 4.1.1 Canvas-Based UI

Recent versions of the Unity engine employ a new way of representing the UI in a canvas. This system is very flexible and offers many specialized effects, as well as automatic scaling based on the size of the screen. Our first versions of the UI for BushBurg utilized the canvas system. However, as time went on, we found our inexperience with the canvas to be problematic. Segmented attributes could not easily be represented with the Slider system within the canvas. They often would not show the correct number of segments, or the segments would be skewed in some way. Additionally, it was difficult to place icons accurately within the UI.

During the first Milestone, we decided to change our UI to a more rudimentary object-based system. This required more coding, and suffers some flexibility problems, but was much easier to make work specifically the way we desired.

### 4.1.2 WorkStation Inheritance

Inheritance is a double-edged sword in game development. On one hand, polymorphism is the key to avoiding a great deal of unnecessary code. Strongly typed languages can be irritating to work with when it comes to declaring variables which could be two or more different types of objects. On the other hand, inheritance is difficult to manage and keep track of while the base class is actively changing.

We knew going into development that a period of refactoring would be necessary to change the organization of our code. It was at this time that an inheritance structure for workstations was introduced. The result is not optimal, but workstation scripts have been kept to a manageable size, and needless sets of if-else statements to declare variables were avoided.

### 4.1.3 Mental Inertia

One unexpected problem we encountered through development was the split of focus between this project and other classes. Unity development is highly dependent on library functions and components. Spending one to two hours at a time within the environment was not typically productive, because most of that time was spent trying to remember how to do basic procedures.

In order to combat this later in development, we began to make more functions in the Utilities class which shortened component calls and common procedures. It is recommended to use this kind of method to simplify code to more familiar terms in future iterations. Static classes like Utilities are one possibility, but it is also possible to actually add extension methods to the GameObject class itself, which accomplishes the same goal.

## 4.2 Balance Problems and Recommendations

Problems with code are often the tip of the game development iceberg. Game mechanics, in the design phase, are essentially a hypothesis. Implementation and testing will tell you whether the design choices were good ones. That is the core function of a systems prototype. Through testing, we were able to identify a few key problems with the attribute, task, and buff systems implemented. Below is a list of those problems and how they might be improved upon in future iterations.

### 4.2.1 Crops, Attributes, & Fatigue

Initially, each task was given a somewhat randomized set of primary, secondary, and recovery attributes. A spreadsheet was used to ensure that all attributes were evenly represented, and that no tasks had identical attribute assignments. Some effort was made to have a logical distribution of attribute importance per level as well. In other words, not all tasks that involve strength are available within the first level, etc.

Each crop has attributes as well. There are individualized fatigue rates, recovery values, and monetary values associated with a given plant or animal. Some plants are considered cash crops, which are best grown to sell. In our current win-condition, these were the most important crops, and are typically sugary. Some crops are good for their restore value. Animal products and gourds were the best plants for this purpose. Some plants simply had specific buffs which were valuable. A detailed spreadsheet of these values and their relationships is located in Appendix A.

Generally speaking, this is a good way to go about plant design, but some of the values in the current iteration are suboptimal or create balance issues. A description of the problems encountered, in regard to the statistical distributions of attributes and crops are listed below.

* Acumen is too important: Most cash crops, as well as some of the animal products and trade task, use acumen as a primary or secondary attribute. It was very difficult to keep citizens’ acumen recovered. In future versions, this attribute needs to be rebalanced.
* Quality attributes don’t feel useful in cultivation: The quality of crops does not matter in regard to monetary value, which means that strategies that maximize use of cash crops will target efficiency instead. This is fine, but it is difficult for a player to distinguish a 20% change in quality of a grown crop. The difference is actually meaningful, but it just doesn’t feel that way.
* Quality calculations are made upon production: Currently, the algorithm for calculating item quality is only done upon completion of the task. For cultivation, this has two problematic implications. First, users can swap out low-quality citizens at the end of the cycle and still maintain quality. Second, the use of a high-quality citizen during production of a crop with a long timer means that fatigue will have more effect on quality than it should. Consider reworking the way that quality is calculated so that time spent during cultivation is important.

### 4.2.2 Problems with Buffs

The buff system is not intended to stay the way it is currently implemented. Recipes will greatly change how buffs are managed, and it is advised to make them more nuanced at that time. When buffs only affect one attribute in one way, many buffs are simply useless.

One particularly useless buff is the Attribute Scalar type of buff. Because of the way efficiency is calculated, we predicted that scalars might have some very specific but incredibly useful applications. A citizen with high base attributes, given a scalar buff, could greatly reduce the time required to perform a task. However, citizens with high attributes are not available early, and never will be. Until the point at which citizens with high attributes exist, those buffs are worth less than simply selling the product. The scalar buff may be viable to some extent, but it should not exist with early foods or recipes because it is of little use at that time.

Choosing one attribute to modify at a time with a buff means that the corresponding attribute in the other type (efficiency vs quality) will disproportionally suffer. Employing a buff to reduce the drain of focus, for example, will keep quality high for a longer period. However, the corresponding efficiency attribute will continue to drain as normal, and the citizen will be pulled from the task anyway. Consider buffs that may be task oriented, or that affect attributes in a different way.

### 4.2.3 A Note on Testing Procedures

Testing within gaming is much different than with traditional software. Automated ‘test cases’ are of little use, simply due to the volume of possible interactions and what it means for an interaction to be ‘successful.’ New game mechanics simply must be tested by hand. In the gaming industry, there is a career for those who can do this well, called Quality Assurance. We did not have access to QA testers, and were forced to spend many hours testing the game ourselves. We are able to give a few recommendations on procedure as a result of our experience.

Create an extension method or static method that will display debug information dynamically. Printing to console within Unity is cumbersome, time consuming, and error prone. It will take some initial effort to set up, but having a debug interface, or floating debug text that can be called easily, is very much worth the time. In our prototype, one such method exists in Utilities. We used a CreateFloatingText method for feedback information during events within the game. One of the parameters of that method call colored the text yellow if it was debug text. Then, if we needed to remove the debug text, we could do a find command in the code for ‘DBUG’ to remove it. There are many examples of dynamic debug windows that are more elegant, but this served us well.

Set up exceptions within your code. When certain things are not supposed to happen, it is prudent to set up your own error messages. Null references can be difficult to track down otherwise.

# 5. Summary

This section contains an administrative summary of miscellaneous personnel and labor statistics, as well as some personal thoughts about the status of the project upon completion.

## 5.1 Labor Report

* Gameplay design & balance – Matthew
* Gameplay testing – Matthew & Khangal
* Client interfacing – Matthew & Khangal
* Client meeting notes 1,2 – Matthew
* Client meeting notes 4 – Khangal
* Weekly team notes – Matthew
* Presentations – Matthew
* Final Report Section 2 – Khangal
* Final Report (other sections) – Matthew
* Version control – Khangal
* Issue tracking – Khangal (mostly)
* Game architecture – Matthew
* UI version 1 – Khangal
* UI version 2 – Matthew
* Game icons – Khangal
* Pause functionality – Khangal
* Other scripting functionality – Matthew

## 5.2 Risk Report

Development of a video game is inherently more risky than traditional software. This is because there is no guarantee that fulfilment of software requirements and objectives will achieve a desirable outcome. As an example, consider word processing software. It is obvious what the design goals of this software will be, and why specific elements, such as spell-checking, should exist within it. As long as these elements are developed to specification, the goal of the software has been met. With games, however, the goal is much more abstract. Design a compelling gameplay experience. The rules of play themselves, that is, the list of requirements of implementation, are somewhat experimental. Even if they are implemented perfectly, those rules may have been incompatible with a ‘fun’ gameplay experience to begin with. There is no way of knowing this for sure until the ideas have been tested within the game. This means that game design is highly iterative. The requirements will surely change over time, and there is no way to predict the exact risks that await us. There are some *categories* of risk, though, that can be described.

Gameplay elements and UI elements are unable to be easily integrated.

User interface elements draw on data from the game during runtime. This means accessing variables or methods with return functions in order to maintain updated information. This did not turn out to be much of a problem as coding standards were set early on.

The UI system is unable to display required information in an accurate way.

There are several implementations of a user interface available to us. It is hard to know, ahead of time, which system is most appropriate. Canvas-based UI elements, for example, scale automatically with screen size and have high versatility in terms of text. Object-based UI elements require more work and don’t scale as easily with screen size, but have the advantage of being very precise. We initially tried to use a canvas

It is not possible to reconcile the mechanics with the constraints of the platform.

We developed and tested the prototype through a Windows PC developer environment. While conversion to a new platform is relatively simple on the software front with Unity, conceptual mistakes could lead to points of no return in terms of small touch-screen interactivity. We do not believe that this is a problem with the current systems prototype. The camera would need a smaller field of view and be mobile, but there are no barriers of our design that would impede that transition.

The core mechanical system implemented is inherently uninteresting or unbalanced.

The first milestone was an experiment of sorts. There was no way of knowing whether our system of rules would provide the experience we wanted. However, with some changes described in the implementation issues section of this report, we believe that the systems that exist are compelling enough to be the basis of a game.

Scripts become too difficult to manage and navigate over time.

Because we didn’t have a rigid system skeleton to rely on, code structure needed to be reconsidered at various points during progress. As a result, a planned period of refactoring occurred after the first milestone. At that point, we had a better idea of ways to manage inheritance, polymorphism, and the relationship between objects. Attempting to do this in advance would most likely have been counterproductive, as we cannot reasonably model the relationships between mechanics we have not yet decided to use.

Our implementation does not meet the expectations of the client.

Early discussion about gameplay ideas is abstract, and there are some assumptions made by the team. It is possible that the direction of the game will be misunderstood on our part, leading to a set of mechanics that, while feasible, are not the desired outcome in terms of the client. As of now, we do believe that this risk was avoided, as Amapalo has shown a favorable response to our demonstrations. The final meeting has yet to take place, however.

## 5.3 Schedule of Events

Early on, we hoped to be able to do a lot more in terms of implementation that we actually ended up doing. The mental inertia of small work windows mentioned in the implementation had a lot to do with it, as well as UI conversion problems. Roughly, though, the work mirrored our expected schedule.

* Early September: Set up version control, meet with client to discuss the abstract ideas of the game, and create a proposal. (accomplished on schedule)
* Mid September: Decide upon prototype mechanics, investigate UI choices, and develop a project plan. (project plan was delayed until October)
* Late September: Meet with client again to discuss refined ideas and plans. Begin implementation of mechanics and UI systems. (accomplished on schedule)
* Early October: Implement milestone 1 and test it. (accomplished on schedule)
* Mid October: Meet with client to discuss the results of milestone 1 and where to go from there. Prepare in-class presentation (accomplished on schedule)
* Late October: Refactor milestone 1 to prepare for long-term development (accomplished on schedule)
* Early November: Design milestone 2 mechanics. (delayed)
* Mid November: Implement and test milestone 2 mechanics, integrate with UI. (partially completed, some features were delayed)
* Early December: Develop new features as time allows (few features were introduced here)
* Mid December: Prepare report, presentation, and finalize code (accomplished on schedule)

## 5.4 Lessons Learned

Matthew

Having had some experience with Unity and game programing, I knew understanding the object-oriented paradigm was important. Having a CS background is fundamental to creating good game code. However, I never really realized the importance of polymorphism in object-oriented software until this project. Transitioning into a polymorphic structure for workstations greatly accelerated development and made everything easier to work with.

This project also has given me a much greater appreciation for the process of software engineering. While the team was small, we still needed to plan for specific events that were months away. Interaction with a client has shown just how important it is to be able to put software ideas into words. Version control software, which I had never used prior to this project, will probably be used in every development project from here out.

Aside from polymorphism, other aspects of the game programming track here at UNO were valuable in the development of this project. Design principles from Art3160, as well as programming conventions learned in CSCI3510 were extensively employed to make the process more streamlined. Practicing these skills in an applied setting should greatly enhance my productivity as a game programmer.

Khangal

Things I learned during the project:   
1.    Learned to use Git version control on Unity project  
2.    Learned to draw use case diagram, and flowchart diagram for game  
3.    Learned to differentiate functional and non-functional requirements  
4.    Learned to use C# dictionary

# 6. Conclusion

Amapalo, our client for this project, has some lofty ambitions. They aim to forge a new genre of mobile gaming and inspire a new generation to give to charity. They aim to build a game company from nothing, and raise money for those in need in the process. The benefits to a game which makes donation to charity a fulfilling and engaging experience for young people has enormous societal benefits.

Of course, none of those goals can really be attributed to our efforts. If BushBurg is a home, we have laid the first few bricks. Instead, our goals were much simpler:

* Guide Amapalo through the game-development process and help them to understand what it entails.
* Come up with a reasonable starting point for systems development, and present a working prototype to evaluate.
* Advise Amapalo in their next few steps on the long journey of making a game.

Hopefully, we have succeeded in those goals to some end. Even if the code itself is unused, Khangal and I have learned valuable expertise. With some luck, Amapalo will benefit from our efforts in their journey to change the world.

# Appendix A: Crop Statistics

Occurrence of attributes as primary/secondary in tasks

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Str | Dex | End |  | Perc | Foc | Acu |
| primary | 7 | 7 | 8 |  | 8 | 7 | 7 |
| secondary | 8 | 7 | 7 |  | 7 | 8 | 7 |

Occurrence of combinations of attributes in tasks

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| combo | | count |  | combo | | count |
| Str | Dex | 3 |  | Perc | Foc | 4 |
| Str | End | 4 |  | Perc | Acu | 4 |
| Dex | Str | 4 |  | Foc | Perc | 4 |
| Dex | End | 3 |  | Foc | Acu | 3 |
| End | Str | 4 |  | Acu | Perc | 3 |
| End | Dex | 4 |  | Acu | Foc | 4 |

Combinations list

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | prim | sec |  | prim | sec |
| Cooking | Dex | End |  | Foc | Perc |
| Trading | End | Dex |  | Acu | Perc |
|  |  |  |  |  |  |
| Yam | Str | End |  | Foc | Acu |
| Corn | Dex | Str |  | Perc | Acu |
| Collard | End | Dex |  | Perc | Foc |
|  |  |  |  |  |  |
| Okra | End | Str |  | Acu | Foc |
| Pumpkin | Str | Dex |  | Perc | Acu |
| Agave | Dex | Str |  | Foc | Perc |
|  |  |  |  |  |  |
| Cabbage | Str | End |  | Perc | Foc |
| Pepper | Str | Dex |  | Foc | Acu |
| Egg | Dex | End |  | Acu | Perc |

Combinations list continued

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | prim | sec |  | prim | sec |
| Squash | End | Dex |  | Perc | Acu |
| Banana | End | Str |  | Acu | Perc |
| Honey | Dex | Str |  | Acu | Foc |
|  |  |  |  |  |  |
| Lettuce | Str | End |  | Foc | Perc |
| Bacon | Str | Dex |  | Acu | Foc |
|  |  |  |  |  |  |
| Pineapple | Dex | End |  | Perc | Foc |
| Milk | End | Str |  | Foc | Acu |
|  |  |  |  |  |  |
| Tomato | End | Str |  | Perc | Acu |
| Beef | Str | End |  | Acu | Foc |
|  |  |  |  |  |  |
| Sugar | Dex | Str |  | Perc | Foc |
| Peanut | End | Dex |  | Foc | Perc |

Effectiveness metrics (Red = bad, blue = good)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | fatigue | time | restore | value | as restore | as value | est. power |
| Yam | 1 | 20 | 0.5 | 10 | 0.000625 | 0.25 | 0.3125 |
| Corn | 1.25 | 25 | 1 | 16 | 0.00128 | 0.32768 | 0.45568 |
| Collard | 1 | 15 | 0.5 | 6 | 0.001111 | 0.16 | 0.2711111 |
|  |  |  |  |  |  |  |  |
| Okra | 1.75 | 30 | 0.5 | 24 | 0.000159 | 0.365714 | 0.3815873 |
| Pumpkin | 1 | 60 | 2.25 | 20 | 0.001406 | 0.111111 | 0.2517361 |
| Agave | 3 | 40 | 1 | 42 | 0.000208 | 0.3675 | 0.3883333 |
|  |  |  |  |  |  |  |  |
| Cabbage | 0.5 | 25 | 0.6 | 10 | 0.001152 | 0.32 | 0.4352 |
| Pepper | 1.5 | 55 | 1.25 | 40 | 0.000344 | 0.352617 | 0.3870523 |
| Egg | 1.5 | 75 | 3 | 60 | 0.001067 | 0.426667 | 0.5333333 |
|  |  |  |  |  |  |  |  |
| Squash | 1 | 45 | 1.75 | 20 | 0.001512 | 0.197531 | 0.3487654 |
| Banana | 2 | 30 | 1.6 | 22 | 0.001422 | 0.268889 | 0.4111111 |
| Honey | 2.25 | 75 | 1 | 90 | 7.9E-05 | 0.64 | 0.6479012 |
|  |  |  |  |  |  |  |  |
| Lettuce | 0.6 | 45 | 1.25 | 20 | 0.001286 | 0.329218 | 0.4578189 |
| Bacon | 3.75 | 55 | 3.25 | 70 | 0.000931 | 0.431956 | 0.5250689 |
|  |  |  |  |  |  |  |  |
| Pineapple | 1.75 | 65 | 1.75 | 60 | 0.000414 | 0.486898 | 0.5283178 |
| Milk | 1 | 80 | 3 | 60 | 0.001406 | 0.5625 | 0.703125 |
|  |  |  |  |  |  |  |  |
| Tomato | 1.25 | 80 | 2.25 | 70 | 0.000633 | 0.6125 | 0.6757813 |
| Beef | 1.55 | 115 | 5 | 110 | 0.00122 | 0.59028 | 0.7122386 |
|  |  |  |  |  |  |  |  |
| Sugar | 3.25 | 45 | 0.75 | 70 | 8.55E-05 | 0.744539 | 0.7530864 |
| Peanut | 0.5 | 150 | 1.25 | 90 | 0.000139 | 0.72 | 0.7338889 |

# Appendix B: Setup

Instructions for PC: Locate the BushBurg\_PCbuild folder, and launch BushBurg\_PCbuild.exe. On the launcher screen, set resolution to a 16:9 resolution (1600x900, 1920x1080) and choose windowed, then launch.

Instructions for Mac: Locate the BushBurg\_MACbuild folder, and launch BushBurg\_MACbuild.exe. On the launcher screen, set resolution to a 16:9 resolution (1600x900, 1920x1080) and choose windowed, then launch.

To examine code: Code is located within the Project\_Code folder, and within BushBurg\_Final/Assets/Scripts.

# Appendix C: Team Meetings

Weekly meetings : Sep6-12

Tuesday, Sep 8

**Discussion Points**

This was a somewhat unprepared discussion about organizational topics. Times for regular meetup (Tuesday/Thursday from 12:00 – 1:30) were established. We talked about our past experience and what each of us would be good at. We had not yet met with Jennifer, so we didn’t have a lot of direction yet.

**Assignments for Tuesday the 8th:**

Matthew – Develop questions to ask the client on Sep9 meeting

Khangal – Set up version control and tracking accounts

Meeting notes for Sep9 meeting with client are in the Client Meetings folder.

Thursday, Sep 10 – Khangal out of town, no meeting

Weekly meetings : Sep13-19

Tuesday, Sep 15

**Discussion Points**

In gearing up to start coding work, we decided to discuss what part of the game we would implement for the first milestone. It was decided to tackle the civilian attributes first as they would lay the foundation for other systems and could be tested in isolation as a sort of game. The guidelines we decided to follow are listed below

Overview

* Current goals for play
  + maximize money gain over time
    - money gain = agriculture output – needs
    - no way to lose
    - fixed prices for now (because of fixed stats/no market forces)
  + Get to level 9 as fast as possible
    - each level has a simple money cost
    - complexity increases with level
* Basic cultivation
  + Statistic-based
  + No seeds for now
  + ‘Slotted’ individual plots
* Basic cooking
  + No recipes just yet
  + Prepared food > raw food of same type
  + Nutrition will be dealt with later

**Assignments for Thursday (for both of us): Considering these guidelines, think about a plausible set of mechanics that we could implement and test.**

Weekly meetings : Sep20-26

Tuesday, Sep 22

**Discussion Points**

It was here that the first milestone for coding and design was set. We discussed several possibilities of implementable systems, as well as ways to represent them in game. A close representation of the decisions we made is available in the milestone 1 gameplay abstract. Mechanics included:

* 9 attributes per civilian, fixed for this milestone
  + 3 efficiency attributes which improve how fast tasks are completed
  + 3 quality attributes which improve the results of tasks
  + 3 overarching statistics that affect the drain/recovery of the task oriented attributes
* Around 20 possible crops to plant, included in the Bushburg\_Crop\_Levels document
  + Each plant can be cooked for specific buffs
  + increased quality improves the buff effect
* Leveling up has a fixed cost, which increases civilian count/crop choice/number of plots
* Crops can also be sold, which is the primary means of ‘winning’ the scenario

**Assignments for Thursday the 22nd:**

Matthew – Create an organization shell within the Unity editor to place new code into

Khangal – Put together icon assets to distinguish the 20 or so crops in the game

Thursday, Sep 24

**Discussion Points**

While the gameplay goals were now clear, much was left to be done in the way of implementation. With the code shell in place, we were now able to start work on the project in earnest. The discussion today was about which aspects of development we would each be responsible for. The tentative breakdown of work is as such:

Matthew

* Design work (non-coded gameplay mechanics)
* Mechanical system implementation

Khangal

* User Interface creation
* Asset work (non-coded art assets)

**Assignments for Tuesday the 29th:**

Matthew – Begin work on mechanical system implementation (structure of civilians, how they can be assigned, etc)

Khangal – Begin work on User Interface for milestone 1 test game

Weekly meetings : Sep27-Oct3

Tuesday, Sep 29

**Discussion Points**

Progress on the first milestone was discussed. About 30% of the needed coding to finish milestone 1 was complete at this point. The topics covered were specific, technical, and in no particular order. Simply put, the minutia of implementation was informally discussed for about forty minutes. We were now preparing for a new meeting with Jennifer on the 2nd of October as well.

**Assignments for Thursday the 1st**

Matthew – Refactor implementation for better organization, prepare gameplay abstract for client

Khangal – Continue work on UI, blend with existing implementation of mechanics

Thursday, Sep 24

**Discussion Points**

Khangal ran through the details of the UI and its current functionally. We discussed how the next set of coding changes would affect how it worked. We speculated about what the next client meeting would entail and how to prepare for it. Work assignments were withheld until Friday after the meeting. We also talked about changing our version control/tracking system to one which more effectively meshed with the Unity framework.

**Assignments for Tuesday the 6th:**

Matthew – Develop gameplay effects for crops, and implement the cooking system

Khangal – UI tweaks to account for new gameplay, prepare new version control

Weekly meetings : Oct4-10

Tuesday, October 6

**Discussion Points**

Not much had changed in terms of the code itself between this meeting and the last, so the meeting was short. We discussed the implementation of the cooking system.

**Assignments for Thursday the 8th**

Matthew – Develop statistical numbers for crop balancing

Khangal – Continue work on UI, prepare new version control

Thursday, October 8

**Discussion Points**

I explained how the balancing of crops would work and went through the spreadsheets developed on Wednesday. We talked about the downsides of our current implementation of crop storage and how we might go about fixing it.

**Assignments for Tuesday the 13th:**

Matthew – Implement draggable crop objects, redesign storage

Khangal – UI tweaks to account for new gameplay, prepare new version control

Weekly meetings : Oct11-17

Tuesday, October 13

**Discussion Points**

Khangal ran me through operation of the GitHub version control, and we tested some uploads to make sure it functioned properly. I explained the new storage system and its effects on the UI. We decided to restructure the UI today, due to the strange effects that the canvas system had on segmented bars. The canvas system was declared ‘too irritating to continue using.’

**Assignments for Thursday the 15th**

Matthew – Improve the visual functionality of farm plots and workstations so that it is easy to tell what is happening.

Khangal – Implement new object-based UI with segmented attribute bars

Thursday, October 15

**Discussion Points**

I worked through some struggles Khangal had getting information from other objects. This took up all of the meeting.

**Assignments for Tuesday the 20th:**

Matthew – Begin testing on milestone 1 to troubleshoot early balance problems

Khangal – Implement new object-based UI with segmented attribute bars

Weekly meetings : Oct18-24

Tuesday, October 20

**No meeting**

Thursday, October 24

**Discussion Points**

Preliminary results of testing were discussed, as well as more adjustments to the UI. An issue with respect to launch resolution was discovered, and we talked about how to fix it.

**Assignments for Tuesday the 6th:**

Matthew – Fix problems with screen space and resolution.

Khangal – Implement new object-based UI with segmented attribute bars

Weekly meetings : Oct25-31

Tuesday, October 27

**Discussion Points**

Discussion in this meeting was entirely based around our expectations for the rescheduled client meeting on the 28th. No new assignments were given.

Thursday, October 29

**Discussion Points**

Results of the client meeting were discussed, as well as the changes we need to make to account for the client’s suggestions. I decided to take over UI development.

**Assignments for Tuesday the 3rd:**

Matthew – Fix problems with screen space and resolution.

Khangal – Test the game and offer balance discussion

Weekly meetings : Nov1-7

Tuesday, November 1

**Discussion Points**

Khangal discussed his experience with the game and its balance. We made new plans for milestone 2, including:

* Water gathering and sanitization
* Hazards
* Caterpillars
* Pests
* Housing and construction

**Assignments for Thursday the 5th:**

Matthew – Make adjustments to balance and test milestone 1 further

Khangal – Test milestone 1 further

Thursday, October 29

**Matthew sick: No meeting**

Weekly meetings : Nov15-21

Tuesday, November 17

**Discussion Points**

We discussed strategies for the final stages of the refactor of milestone 1, as well as ways in which we might implement certain facets of milestone 2.

**Assignments for Thursday the 19th:**

Matthew – Finish refactor of milestone 1

Khangal – Develop floating text for debugging and feedback

Thursday, Nov 19

**Discussion Points**

I explained the milestone 1 refactor of WorkStation, which changed the way in which those objects interacted with UI elements and other scripts. Many deadlines were piling up in other classes, so we decided not to meet the following week.

**Assignments for Tuesday the 1st:**

Matthew – Begin work on the implementation of Pests

Khangal – Develop floating text for debugging and feedback

Weekly meetings : Nov29-Dec17

Tuesday, December 1

**Discussion Points**

Recalibration back into focus on the project, primarily. Khangal ran me through the floating text script, and I discussed the pest implementation. Further work was tabled due to deadlines in other classes.

**No Assignments**

Thursday, Dec 3

**Discussion Points**

We discussed the final report and how we might split up work on it. Implementation of milestone 2/3 had seen many delays, with large amounts of work due in other classes. I discussed my plans for improving in-game feedback with floating text, and completing work on pests

**Assignments for Thursday the 10th:**

Matthew – Improve feedback mechanisms through floating text, finish pest and forager implementation

Khangal – Begin work on section 2 of final report

Thursday, Dec 10

**Discussion Points**

In our final regular meeting, we discussed the final report and the presentation on the 17th. Other classes were wrapping up by that point, and I was able to devote more time to the project.

Assignments for Thursday the 17th:

Matthew – Prepare presentation

Khangal – Finish work on section 2 of final report

# Appendix D: Correspondence

**Matthew, Sep3:**

Hello Jennifer!

Khangal and I will be your developers, so to speak, for this project.  I'm a game programming student myself, and will be pursuing independent development after graduation.  There are quite a few ways in which the game idea you've outlined shows potential, and I'm really looking forward to exploring them.

Dr. Siy suggested a sit-down meeting to discuss things like scope and design goals.  I couldn't agree more with this notion; both Khangal and I have many questions about your expectations and how to meet them.

My schedule, at least in the next few days, is very flexible.  Other than late Tuesday morning, anything will work.  Dr. Siy wanted to sit in if possible.  Let me know what works for you!

Thanks for your time.

**Jennifer, Sep3:**

Hello Matthew,  
  
Thanks so much for contacting me.  I am really looking forward to working with you and Khangal!  My schedule is also very flexible.  I would also like to avoid Tuesday morning and also have a conflict 3-6pm each afternoon.  
  
Harvey and Khangal,  is there a time that works best for you?  
  
Thank you,  
Jennifer

**Harvey, Sep4:**

For next week, I have the following available times:  
Wednesday, 9/9: before 11am and between 12-1:30pm.  
Thursday, 9/10: before 11am  
Friday, 9/11: before 1pm

**Matthew, Sep5**

Of the specific times mentioned:

Wednesday 12:00 - 1:30 is best

Thursday morning won't work for me

The rest are fine

**Khangal, Sep6**

Hi All,

I am very sorry about not responding e-mails. I had some technical problem with my laptop. Now, everything is good and I will be more responsive.

For the meeting: I will be **available** on Wednesday  **before 11:00am and 12:00 - 1:30.**

Thursday, 9/10: before 11am (**I have a class**)  
Friday, 9/11: before 1pm (**available**)

Regards,

Khangal

**Jennifer, Sep6:**

Hi everyone,

Let's plan on meeting Wed. 12:00-1:30 since everyone has it available. Dr. Siy where is a good place to meet? If this is during people's lunch hour would everyone like me to bring sandwiches from Jason Deli? What you sandwiches would you like?

Looking forward to meeting with you,

Jennife

**Harvey, Sep8:**

Hi everyone,

Wednesday 12-1:30 is okay with me. I have reserved PKI 172A for this meeting.

Sandwiches sound good! I’m fine with any sandwich.

Thanks,

Harvey

**Matthew, Sep8:**

That will work!

**Khangal, Sep8:**

Any sandwich will be fine with me too. Thanks for that. See you all tomorrow.

**Jennifer, Sep11:**

Khangal and Matthew,

I enjoyed getting to meet you and look forward to this collaboration.

I put pictures in a drop box file for you to see and get the feel for the area. Matthew, you mentioned additional research but if you look at my flow chart for the game, you will see only crops native to that region and hardships related to that area. I don't want to get bogged down in an exact representation. I want it to have the feel of the area. It is from Serenji, Zambia if you want to look some more. If there is anything you want from me,please let me know.

Jennifer Olver

**Jennifer, Sep12:**

Guys,

please also see link to video below on Zambia.

Best Regards,

Jennifer

<https://vimeo.com/139066331>

**Matthew, Sep12:**

Thank you for the material Jennifer!

To clarify, it's not that we are interested in recreating specific details.  In fact, everything will be quite abstract without graphical design intact.  The details help us to think up specific mechanical interactions that could be fun and appropriate to the theme - a sort of brainstorming tool.  It is easier to draw that kind of thing from details of everyday life than from a broad understanding of the region.  The flow chart will go a long way with that.

**Jennifer, Sep23:**

Hi Gentlemen,

I was following up to see how the project plan is coming along.  Also,I was wondering if we needed to get together to review the project plan. Please let me know your thoughts concerning next steps and timing.

Thanks,

Jennifer Oliver

**Matthew, Sep24:**

Hello Jennifer!

In an abstract sense, we are planning to develop in a highly iterative way.  This is fairly close to agile development in other fields.  The basic process is to think about and implement a set of mechanics, then test them to satisfaction. A comparison might be made to doing a science experiment; it's more of a hypothesis than a blueprint.  This process will be done perhaps 4-5 times in the time that we have.  In a sense, this means that there will be 4-5 project plans, because the specific requirements will change depending on the results of the testing.

We are gearing up for the implementation of the first set of mechanics this weekend, followed by 1-2 weeks of testing and balancing.  The system being implemented is the individualized attributes of each 'sim,' or simulated person.  It will be the foundation of cultivation and cooking as well.  I will put together a quick abstract document that details the system this afternoon.  We can schedule a meeting at that point if you prefer to talk about it in person.

**Matthew, Sep24:**

Attached is an informal document which describes our first system implementation.  It is nontechnical.  Let me know how you feel about this, and whether we need to meet to discuss it further.

Additionally, the current working list of crop unlocks per level is here as well.  I moved some of them around from the flow chart in order to have a consistent rate of unlocks.  Let me know if this is a problem.

**Jennifer, Sep28:**

Matthew,

I love what you are doing with the game and seeing my vision starting to take shape. I looked at both documents.  I love to see the authentic Zambian crops in the game. The order doesn't matter to me but thank you for laying it out so well.

Milestone 1 obviously has had a lot of work on it and you expressed the concepts of the game very well. I thought your bottom line on how the game will played really captured the essence. I also liked how you spelled out the attributes of the "sim" characters and their "tasks". I know you are still in first implementation stage of the game but in bringing the vision I had to life I have a few questions and aspects of the game I would like to be considered. I think it's better for the three of us to meet in person and discuss. Friday is completely open for me if you and Khangal have any time available.

I don't want to interrupt your project or cause you to get behind, so I will lay out a few of my thoughts I would like you to consider. I would like to explore the possibility of a hazard/vulnerability/risk attribute or global attribute added. I think making those hazards an integral part of the game helps identifies it specifically to Africa and adds to the distinct flavor of the game.  It also adds in educating players of the realistic challenges a person faces in that area of the world. I thought it would be fun for the game to show the outdoor cooking verses a stove but I didn't understand the importance of raw food being eaten compared to cooked food.

Thanks for all of your hard work and expertise in bringing this concept to life. Please let me know if you and Khangal are available Fri.

Jennifer

**Matthew, Sep28:**

I'm glad you approve of the first system, and I hope we can make it fun to interact with over time.  If the order of crops doesn't matter, we may change it around more later on.  It would be nice if we can make some nutritional analog in terms of the effect of each food.  This would make everything a little more intuitive.

I may have entirely neglected to mention this in the abstract I sent, but keep in mind that this milestone is not meant to be holistic.  There are plenty of intended functions that are simply not going to be introduced at all right away, because we want to test an isolated system first.  Things like housing, local trade, training, water and sanitation quality, etc.  Environmental hazards are among the features that are eventually going to be implemented, just not for this milestone.  Conceptually, diseases could be represented as negative modifiers the same way that food represents positive modifiers.  For example, tuberculosis might impose a cap of 60% health and nullify the endurance attribute.  That effect would last until treated by medicine.  We can talk more about this.

I am available this Friday to meet.  For future reference, it is far better for me if the meetings are Monday-Thursday as I do not live in Omaha.  For a one time thing, this Friday is totally fine at any time of day.   Is there anything you'd like us to prepare for it?

**Khangal, Sep29:**

Dear Oliver,

Thanks for your feedback on current progress.

I am available before noon on Friday. Since, both you and Matthew are available on Friday, Can we meet at 10:00am on Friday at PKI?

Regards,

Khangal

**Jennifer, Sep29**

Khangal,

I can make 10:00am at PKI if that works for everyone else.

thanks,

Jennifer

**Khangal, Sep30:**

Dear Jennifer, Matthew and Dr.Siy,

I booked a room (278) for our meeting at 10:00am at PKI.

Khangal.

**Harvey, Oct1:**

Hi everyone,

I was also able to book a conference room for you guys. It is PKI 150. Would that be preferable?

Thanks,

Harvey

**Harvey, oct 1:**

FYI, looks like the dean’s office went ahead and canceled the PKI 278 reservation. So you have PKI 150. I’ll drop in about 10:50 to talk about the IP transfer paperwork.

Harvey

**Matthew, Oct1:**

Okay Dr. Siy, thank you for your help.

Jennifer, Khangal, and possibly John - see you tomorrow at 10:00 in PKI150.

**Khangal, Oct1:**

See you all at 10:00 am tomorrow in PKI 150.

**Matthew, Oct3**

Hello Jennifer--  I hope our meeting on Friday was of some use to you.

You offered to do some of the nontechnical research for the game, and there is definitely some research that can be done for milestone 1.  It is described below.  Let me know if this all makes sense.

One way to help players internalize the mechanics of food and cooking is to give the foods a real world analog.  Since we are using real food types, it would be nice if the buff effects made intuitive sense.  For example, most people that are trying to build muscle mass will intentionally consume a lot of protein, because this helps to repair damaged muscle tissue.  To that end, foods that are valuable for their protein content might be more balanced toward the improvement and recovery of the strength attribute.

Attached is the current list of crops/animal products that will be in the game.  There are two lines of research that could be very useful for me in setting up which foods have which effects.

Line 1: A nutritional profile on each food.  It doesn't need to be complicated.  Simply the top 1-2 nutritional benefits of that food. Some might be good for their fiber, some for a specific vitamin, etc.

Line 2: For each 'nutritional benefit' from Line 1, an explanation of what that benefit actually is.  As an example, Vitamin A is used by the body for immune response and vision.

Thanks for your help!

**Jennifer, Oct7:**

Yes, Friday helped a lot. Thank you. Sorry I had not replied sooner.  
  
I’m almost done with the research and will get it to you tonight. Hopefully with recipes. Anything non-technical I can help you and Khangal with I would be glad to do.  
  
Jennifer

**Jennifer, Oct7:**

**I missed this but we have to have corn/maze.  corn is a foundation/staple food. It should be in the Level 1.**It is the big white stuff in the pictures I gave you.

Level 1   should have caterpillars for protein. Available only half of the year during summer

**Sweet potato**,  should be Yams instead-fiber. Vitamin A –keeps eyes, skin, teeth and bones healthy.

**Agave**– it’s a sugar. It gives energy but no nutrients. I would take this out of level 1. maybe collard greens or Okra here instead?

Level 2

**Okra**– fiber, Vit.C, A, K healthy pregnancy, healthy digestion, stabilizes blood sugar

**Pumpkins**- beta carotene, anti-oxidant vit. A, C, E, also has seeds which are good for protein and brain health, helps eyes and skins

Level 3

**Collard greens**– Vit. K, good for bones, anti-cancer, high fiber and water for healthy digestion, sleep and mood, maybe make them happy?

**Squash –**very similar to pumpkins

**Chickens –**protein, all vit. B’s, immune, energy, heart and brain health, don’t require a lot of food or space to raise,

**Eggs-**protein, strong muscles, immune system,

Level 4

**Sugarcane – water bugs –**simple sugar but good to trade

**Banana-**vit. A, potassium, energy, good for eyes and bones

**Bees/honey-**energy, anti-bacterial and anti-fungal used as a natural antiseptic

Level 5

**Cabbage-**vit. K, fights inflammation, has probtoics

**Peanuts-**protein,growth and development of the body,sustained energy

**Pigs-**protein, need more space to raise and require quite a bit of food to feed

Level 6

**Pineapple-**vit. C, immune support, energy

**Goats**- require a lot less food and space to raise than cows,

**Milk**- Vit. D**,**calcium for growth and strong bones, proteins

Level 7

**Tomatoes-**lycopene for anti-oxidants, Vit. C, A, potassium

**Cows-**same kind of benefits as cows but cost more, need more space to raise and more food to feed them

Level 8 lettuce has no nutritional benefit  ,

**peppers** are close to tomatoes

Recipes I will send these by Friday. My assumption is that you want them to start with a few ingredients and with lower level ingredients and add from there.

Hope this helps,

Jennifer

**Matthew, Oct8**

Thank you Jennifer, that will be very useful when setting up the buffs.  This process will almost surely lead to changing the unlock order around again.

Recipes are definitely a thing we'll need help with, though it is not time sensitive; recipes probably will not see any implementation until November.  You are correct on the complexity assumption.

**Jennifer, Oct14:**

Khangal and Matthew,

I am sorry but I will be unable to come tomorrow and see the end work of phase one. I coach cross country and they moved the time earlier for tomorrow’s meet.

Jennifer

**Khangal, Oct14:**

Jennifer,

No worries, we can set up another time next week.

Khangal

**Matthew, Oct26:**

Hello Jennifer,

It occurs to me that we never actually scheduled a meeting with you to demo milestone 1 after the cancellation.  Are you able to meet this week on Wednesday(28th) or Thursday(29th) at 12:00 - 1:30?

**Jennifer, Oct26:**

Hi Matthew,  
  
Wednesday the 28th at noon works great for me.  
  
Jennifer

**Matthew, Oct26**

Okay, great!

Dr. Siy, can you book us a room for Wednesday the 28th at 12:00?

**Harvey, Oct27:**

Matt,

I’ve sent the reservation request. Will let you know when I hear back.

Harvey

**Harvey, Oct27:**

Matt,

You can use PKI 172A tomorrow from 12-1.

Harvey

**[Entry from Jennifer lost to autodelete in spam folder]**

**Matthew, Nov5:**

Hey Jennifer,

Sorry I haven't responded to your request for a Mac build.  For some reason that email thread is routing to our spam folders.  I think this is the first time that has occurred for me on this unomaha account.  Khangal made me aware of the problem today in our meeting; I will run a mac build and test it tonight when I get home.

**Matthew, Nov5:**

There is a new folder in the google drive called 'macbuild'

You will need to download the whole folder for it to work.  You may get warnings about security depending on your system settings.  This is because the file is a .exe, and runs code when you open it.  Apple has a sort of certification process for applications- all applications which have not gone through this will be flagged as possible viruses.

Let me know if there are any further problems!

**[Entry from Jennifer lost due to autodelete from spam folder]**

**Jennifer, Nov23:**

Matthew and Khangal,  
  
Thank you so much for all of your hard work on the game. My family and I have been playing it and enjoy all of you hard work. I hope everything continues to go well. I can’t wait to see what else you are able to get done. I know you are getting close to finals so you will be winding down soon.  
  
I was wondering about a few changes. Can three of the people at the beginning of the game be much lower on their attributes? I would like them to represent children. If they are children, it makes it much easier to send them to school in later levels and hire others with higher attributes to replace them. Also, when we get to higher levels, can we add more cooks? We found it hard to keep up.  
  
Thanks so much,  
Jennifer

**Matthew, nov23:**

Hey Jennifer,

The functional distinction of children, and aging for that matter, were not things I expected to tackle in this prototype, but simple changes like that are totally feasible.  If you prefer, we can reduce the size of citizens that are regarded as children as well.  Training and schooling are mechanics we intend to get into soon.  Additional benefits and balancing to cooking are also good ideas, and we'll look into them with the newer versions.

While we're talking about future changes- now would be a good time to start planning recipes.  If you are still able to do some conceptual work with that, it would be a great help.  In an ideal case, each ingredient would be a component in 2 recipes, and at least one recipe would be unlocked every level.   Those might be difficult quotas to meet.