

# Cradle of Civilization – Final Proposal

Team 4: Jeffrey Lym, Michael Leece, Katie Ewing, Russell Melick

## High Concept

The Cradle of Civilization is a turn-based strategy game set in the time of the Agricultural Revolution. The player begins as the leader of a tribe of hunter-gatherers, but quickly realizes that this method of survival is insufficient to meet the tribe's needs. The player then must successfully lead their tribe through the process of choosing a viable location to settle, where they will be able to generate a stable food supply. Once found, the player will need to turn their energies towards building a thriving settlement. The tribe will be forced to overcome natural disasters, and branch out to trade with other villages. Eventually, through skillful management of resources and labor, the player will build their settlement into a city that is the birthplace of a civilization.

## Key Requirements

These key requirements were obtained by examining our initial problem statement and by talking to the teachers from Kalamazoo. The first three are the most important; the game simply will not be useful if any are not met. The final two are not critical but are tied to the quality of the game and how appealing it will be.

*Workable:* The game must run on the school's computers to be useful. It must run on Windows, Hewlett-Packard personal computers.

*Convey Key Concepts:* The game must create an experience that can be used as a framework for explaining the impact of the Agricultural Revolution. Specifically, it must explore why people moved from hunting and gathering to settling, and that settling resulted in a stable food supply. It should also explore the idea that settling also leads to division of labor and trade.

*Enjoyable:* The game must be fun for sixth graders to play. It should appeal to children with different interests and should be enjoyable if played in twenty minute intervals.

*Intuitive User Interface:* It is difficult to enjoy a game and focus on the learning objective if the user has to fight with a poor user interface.

*Polished Appearance:* To hold the students' attention spans, the game must have visuals and sounds that are competitive with modern games.

## Game Overview

The game begins with the player being introduced to their situation by their friendly advisor. They are the chief of a wandering tribe, whose method of acquiring food is through hunting and gathering, which takes up almost all of the tribe's time. The advisor has heard tales of tribes who have settled in a single location and thrived, and thinks that this may be a valid option to consider.

After hearing what the advisor has to say, the player begins play in the hunter/gatherer phase. In this phase, they are free to hunt, settle, or move their tribe around the map. After hunting in one location, it becomes necessary to move on to find more game, while moving consumes food. It soon becomes clear that hunting and gathering is not a stable enough food supply to support the tribe, and the player must pick a location to settle at or watch their tribe slowly dwindle.

Once a player has made the decision to settle, the game moves into the settlement phase. The player moves to a smaller map, which is representative of the location on the larger continent map that they chose to settle in, and begins building their settlement. They have the option to build various buildings such as houses, farms, and storage pits, which all provide different functions for their blooming village. Without enough houses, the tribe may lose some members during the winter cold. Without enough farms, there may be a lack of food to go around. The player also has control over where their tribe's labor output will be directed, now that there is time to focus on other things than hunting and gathering food.

As the player's settlement grows new dangers open up. Natural disasters keep the player wary. If a horde of locusts descends on your crops, do you have enough food stored away to survive the meager harvest? However, along with these tragedies come new opportunities. Trade can open with nearby settlements, and the option to acquire more advanced technology and methods can improve your tribe's labor efficiency. Eventually, through skilled management of resources and labor, the player's tribe becomes a burgeoning civilization, and victory is accomplished.

## Key Features

The game's features are divided into two categories – features that relate to its hunter/gatherer stage and features that relate to its settling/agricultural stage.

### Hunter/Gatherer Stage

This stage is the first phase of game play and it involves the player guiding their tribe across a large landmass, hunting and gathering food until the player finds an area to settle on. This stage aims to show what people did to survive before the Agricultural Revolution and why they eventually settled. Key features include:

- A map that shows where the player's tribe is, and where food is. It also gives a summary of the geography of the land the player's tribe is currently at. The player can move their tribe around the map to explore, to go to an area where there is food, or to find the best place to settle.
- A set amount of stored food that decreases whenever the player moves their tribe. This illustrates the idea that before the Agricultural Revolution, people did not have an effective means to store food and could only carry a small amount with them. If the tribe runs out of food, their population size decreases after every move. Their people are hungry and some are dying from starvation.
- Hunting and gathering mini-games. When the tribe lands in an area of land that has food, they can choose to hunt or gather (depending on the type of food available), which allows them to play a hunting or gathering mini-game. These mini-games aim to add different types of game play to the game so that it will

better appeal to a wide variety of children. A successful play restores the amount of food the tribe has stored so they can traverse the map without penalty.

- The choice to settle.  
The game is skewed so that the tribe will lose people as it tries to get to the next area with food. This helps to illustrate why the Agricultural Revolution occurred. When the player realizes that they cannot maintain their tribe, they can choose to settle on the land their tribe is currently on. Players will have to move their tribe to lands with a good geography to increase their chances of survival in the next stage.

### **Settler/Agricultural Stage**

This stage contains the main game play where the player builds and manages a village. It aims to show how changes made in the Agricultural Revolution led to more stable societies. Key features include:

- An overview map.  
The map shows the geography of the area, and is divided into a grid with different terrain types such as rivers, fertile soil or desert, as well as showing the player's buildings.
- The ability to build buildings with different functions.  
These buildings include farms and animal pens, storage pits to store food for harder times and huts for villagers to live in. Creating buildings require labor and resources, and their performance can be affected by the labor assigned to them, the terrain that they are built on, and seasonal factors.
- The ability to divide labor.  
After creating an initial food supply, the player will have to learn to balance their labor well among activities such as farming, storage and construction, in order to continue to grow.
- Natural disasters.  
After the player reaches a certain level of stability, natural disasters such as floods, hurricanes and plagues will occasionally appear. This adds an extra layer of challenge to the game. The player will be forewarned of disaster and will have a few turns to gather enough surplus food and build enough shelters. Failing to prepare can lead to much loss.

## Competitive Research

	Summary	Strengths	Weaknesses
<b>The Cradle of Civilization</b>	A turn-based strategy educational game where the player attempts to successfully navigate their tribe through the Agricultural Revolution	<ul style="list-style-type: none"> <li>* Appealing to multiple different groups of people</li> <li>* Good reward system – something to brag about</li> </ul>	<ul style="list-style-type: none"> <li>* Graphics will not be up to par with modern games</li> <li>* No name-brand recognition</li> </ul>
<b>Civilization IV</b>	A turn-based strategy game where the player guides a civilization through the ages in competition against other player or computer civilizations	<ul style="list-style-type: none"> <li>* Excellent Replayability</li> <li>* Multiple paths to victory</li> <li>* Good interface – usually don't have to descend into subscreens or menus</li> <li>* Successful franchise</li> </ul>	<ul style="list-style-type: none"> <li>* Endgames can be very slow</li> <li>* Purely strategic game, appeals to a limited audience</li> </ul>
<b>Caesar III</b>	A real-time strategy game based in Roman times in which the player attempts to build a thriving metropolis, needing to manage food, trade, crime, and a number of other aspects of their city	<ul style="list-style-type: none"> <li>* High level of control over the player's city</li> <li>* Shallow initial learning curve</li> <li>* Realistic and mostly historically accurate</li> </ul>	<ul style="list-style-type: none"> <li>* Forced to micromanage city in order to succeed</li> <li>* Complex (relative to our audience)</li> <li>* Limited number of maps to play on</li> </ul>
<b>Food Force II</b>	An educational simulation game designed to raise awareness about famine and other problems in impoverished countries	<ul style="list-style-type: none"> <li>* Succeeds in goal of teaching about the needs of an impoverished village</li> </ul>	<ul style="list-style-type: none"> <li>* Levels too specific, removes creative and strategic elements</li> <li>* Poor interface, difficult to do simple tasks</li> <li>* Due to specificity, not very replayable</li> </ul>
<b>Oregon Trail</b>	An educational game in which the player attempts to successfully cross the Oregon Trail, while needing to make real-life decisions such as what gear to take, or which branch of the trail would be best	<ul style="list-style-type: none"> <li>* Broad appeal, elements that appeal to multiple gaming styles</li> <li>* Educational aspects ingrained in game, instead of feeling "tacked on"</li> <li>* Simple interface</li> </ul>	<ul style="list-style-type: none"> <li>* Gameplay feels slow at times</li> <li>* Players can become obsessive about a single aspect of the game (e.g. hunting) to the point where they ignore the overall objective</li> </ul>
<b>Carmen Sandiego</b>	An educational game where the player follows a series of clues about countries around the world in order to track down and apprehend criminals	<ul style="list-style-type: none"> <li>* Great writing and personality</li> <li>* Good reward system – Hall of Fame and periodic promotions</li> <li>* Feedback system lets sleuths know if they are on the right trail</li> </ul>	<ul style="list-style-type: none"> <li>* Only teaches trivia, no deeper aspects of countries</li> <li>* Clues sometimes get repeated</li> </ul>

## Summary of Competitive Research

Through the process of competitive research, we have arrived at the conclusion that there is no game currently on the market that fulfills the specific needs proposed by our client. As such, our product will be superior as a teaching tool to existing games, being explicitly tailored to meet the client's objectives. For example, while Civilization has a similar model, our game is more suited to 6<sup>th</sup> graders, with a shallower learning curve and a simpler interface. Also, it is appropriately designed for the amount of time the students will be able to devote to it (about half an hour a day). When nearing the end of some of these games, very little can happen in half an hour, which hurts the feedback loop for the users. Also, our game is more focused on a single settlement, instead of building an empire. Our game will introduce to the students the key concepts of the Agricultural Revolution, in such a way that they will have a concrete idea with which to correspond their in-class learning. At the same time, it will be entertaining and fun, so that the students' minds are engaged by the material, and not distracted. In short, our game is necessary because the Agricultural Revolution is not terribly exciting, and students become turned off by unexciting subjects rather easily.

We also learned several critical game design lessons through our competitive research. For example, it is important to have a good reward and feedback system that encourages the user, and gives them a sense of accomplishment. Having something to brag about to your friends is desirable, and can be an incentive to be that much more involved in the game. Another important lesson was the need for an appeal to a broad audience. We are creating this game for an entire history class, which will have any number of students and play-style preferences. It is important that the game be fun for more than just the strategy players. Something to be wary of is limiting the creative force of the individual player. Games that force a user down a specific path end up being predictable and boring. This is especially key for the learning component of the game, in as much as the students will learn the lessons more efficiently if they come up with them on their own, instead of being forced into them.

## Rationale for Game Design

Our game is meant to be educational and fun for a group of sixth-grade students. Our design successfully gives the players a fundamental understanding of the Agricultural Revolution while staying away from feeling too much like school.

The format of our game allows the player to pick up on many key ideas from the fundamental gameplay. At the beginning of the game, as a hunter/gatherer, the player begins to realize that even with all of their resources focused on hunting and gathering, it is difficult to keep enough food for the tribe. As their food dwindles, players will see the importance of finding good land to settle on, and will choose to begin a life of agriculture. Later, as chief of the village, players will see that the village only grows when there is a stable food supply from farms and domesticated animals, there is sufficient room to store surplus food for the winter and for natural disasters, and when labor is divided in a sensible manner. Many students will notice that their village can grow even faster with the help of trade.

For students who may not be picking up on these key concepts, we have an advisor character. The advisor will help struggling players by alerting them to problems (for example, a food shortage) and suggesting possible ways to remedy the problem. The advisor will also alert the player to new developments, such as the arrival of a trader or the possible flood danger from a rising river. With these two functions, the advisor will help keep the player on track as well as reinforce key concepts.

This game will also be fun for students to play. It will be challenging to create a thriving settlement, and there will be several plausible strategies for getting there. Some students, for example, may invest heavily in making each farm as efficient as possible, by increasing workers and improving the land, while other players may simply choose to make as many farms as they can afford. Students will enjoy experimenting with different strategies to see what will make their village grow. Additionally, we have reward loops of several different lengths, which will keep students engaged over a period of time. A turn-based game means that there is a short-scale positive feedback loop, with changes happening at every turn. Possible rewards, such as prize buildings or technologies for reaching certain targets, and periodic natural disasters will comprise a mid-sized feedback loop. The largest loop will be the overall game objective, which is to grow your settlement to a certain size.

Particularly enjoyable for students, we believe, will be the occurrence of natural disasters. Not only will the disasters make managing the village more challenging, but destruction creates a certain voyeuristic joy in many people. The occasional disaster will keep the gameplay from becoming too monotonous, and will be entertaining for many of the students.

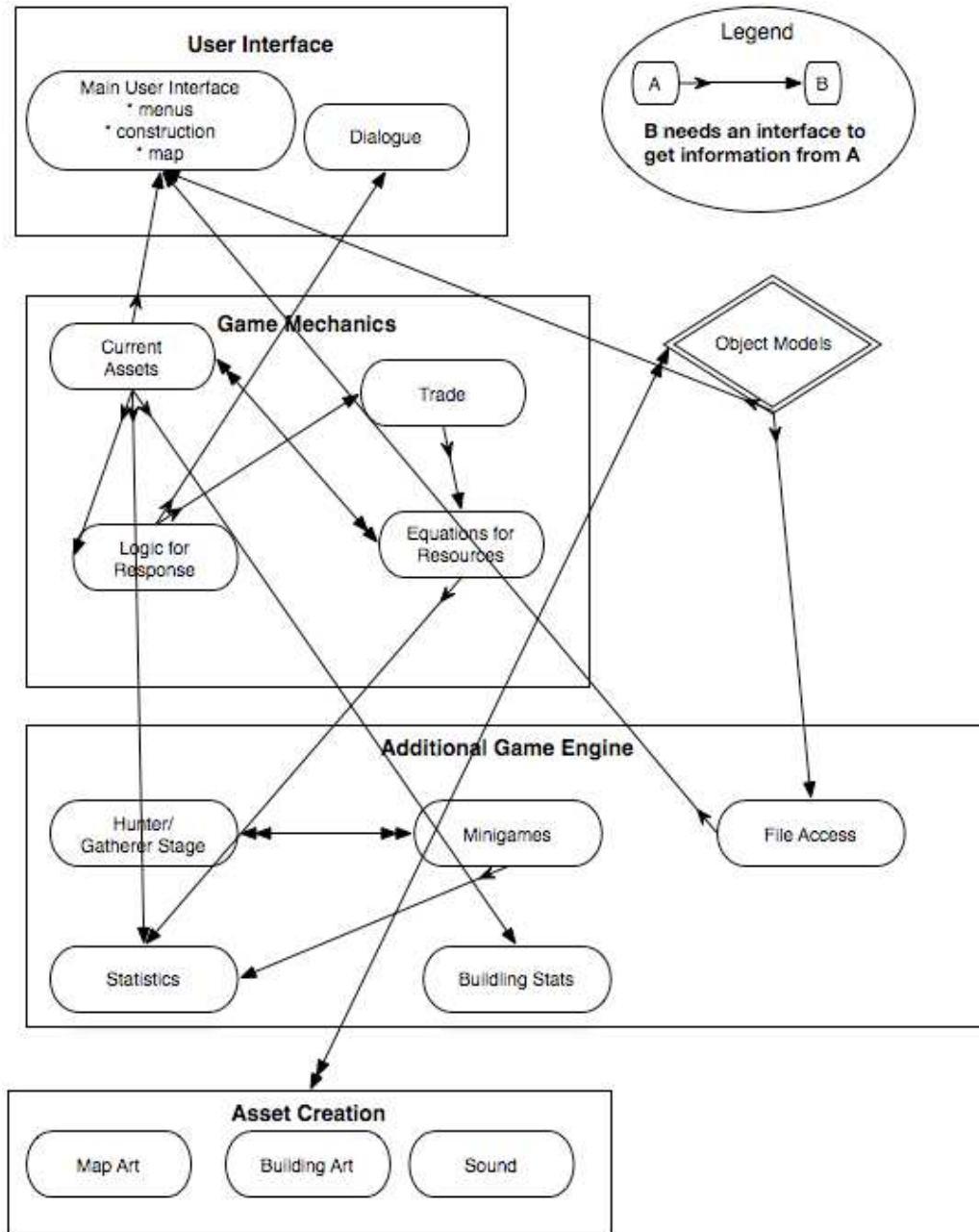
Additionally, we plan to include at least one minigame. The minigames will provide an alternative form of play for students who do not enjoy strategy and simulation games. This will keep these students working to advance their village, and learning the learning objectives, simply to obtain the chance to play the minigames again. However, the minigames other than hunting are optional to the gameplay, and so will not distract students who prefer the main game.

We believe that this game will be both appealing and educational for our sixth-grade audience. By keeping the learning objectives enmeshed with the fundamental gameplay, we make the objectives difficult to miss, while making the game not seem too 'educational.' The type of play and level of challenge will be appropriate for the students, which will keep them having fun.

# Proposed Project Overview

## Architecture

There will be several key components/groups to our implementation. A rough diagram is shown below.



### User Interface

*Dialogue*: handles the feedback the user receives as they play through the game. This ranges from congratulations on new advancements to constructive advice on how to advance their civilization. It requires a good interface to the Logic for Responses component, and access to drawing on some parts of the main game screen. It must interface with the user such that it does not bother them overmuch.

*Main User Interface:* handles several key components for how the user interacts with the game. It will contain the menus (main menu, save menu, and others), the construction interface, and the main map display. The menu will require an interface to the File Access component to save games. The construction interface will require an interface to the Current Assets, and will to communicate with the main map display. The main map display requires an interface to the Current Assets so they can be displayed.

## **Game Mechanics**

*Current Assets:* keeps track of the player's resources and buildings and provides an interface to other components that want to know the values of those resources. It requires good knowledge of the object models. It must interface with the Equations for Resources, Logic for Responses, and the Main User Interface.

*Equations for Resources:* not visible to the user, but controls how the settlement earns and consumes resources, and how their work division and trade affect their resource production. It must interface with the trade and work division areas, and the Trade component. It must provide an interface for the Building Stats component so it can request building efficiency data.

*Logic for Responses:* keeps track of the user's progress and decides when to congratulate them, when to send them hints to help them advance their settlement, or when to enable new trade. It requires a good set of rules for the cases of user notification. It needs a strong interface to the Current Assets so that it can interpret how the user is playing the game. The dialogue component depends on a strong interface with this so that it knows what messages to send when.

*Trade:* keeps track of the player's trade interactions, and offers them new benefits as time passes. It requires interfaces with the Logic for Responses component and with the Equations for Resources component. It must interface with the user interface.

## **Additional game engine**

*Building Stats:* shows information about a building when it is selected. It is almost a graphical representation of the object model. It requires an interface to the Equations for Resources component to query building efficiency data. It requires a strong interface to the object models.

*File Access:* handles all of the reading and writing of save files to disk, to enable the students to continue their game at a later time. This depends on the object models - how we keep track of what buildings they have and where they have placed them, and on the statistics models - how we keep track of their population and resource count. The save and load menus depend on having an interface to this component.

*Hunter/Gatherer Stage:* the first section of the game that has the player wandering the continent searching for food before they decide to settle somewhere. It requires interfaces with Hunting Minigame and with the main game.

*Minigames:* there will be two separate minigames that the user can play. The Hunting minigame will be played during the Hunter/Gatherer stage. It affects how much food the player receives that turn. It requires an interface with the Hunter/Gatherer resources/statistics section. The Farming minigame can optionally be played during the main portion of the game. It does not have any effect on the player's resources. It requires an interface with the Statistics component. It must interface with the main game.



*Statistics:* keeps track of significant events during the users play, such as population history and minigame successes. It requires an interface with the end of game events. It must interface with the Current Assets component and both minigame components.

### **Asset Creation**

*Building Art:* all of the artwork for buildings and other objects placed on the map. It must interface with the object models so that the objects can be drawn correctly. It depends on map specifications.

*Map Art:* all of the artwork that creates the general display map, besides the buildings. It must interface with the object models so that the map can be drawn correctly. It depends on map specifications.

*Sound:* creates the sounds that other components use, whether it is for clicking a button or for background noise to set the atmosphere. It must provide interfaces to play a sound once and to loop a sound as a background.

### **Tools**

We looked at several alternatives to implementing our game from scratch, including an open source game that seems to contain a large amount of our desired functionality. However, without much more extensive research, it is impossible to know how heavily it would have to be modified to fit our concept. Several key components are implemented quite differently than we were planning. Additionally, by starting closer to scratch, we as students will learn more about developing correct architecture and all the steps of creating a game. Once we decided that starting from close to scratch was the best option for our development, we had to choose a development platform. After evaluating many options, including Java, Flash, C++, and others, we decided that the XNA Game Studio framework created by Microsoft was our best option. XNA uses the C# programming language, similar to C++ and Java, which all students in the class have had experience with. Thus, there should be limited down time due to learning a new language. Besides giving the class access to professional grade development tools and environments, XNA has many tutorials online, created by both other users and by Microsoft. Thus, it should be easier for students to begin coding than it would be with other unfamiliar environments.

Our project would use Photoshop as the main graphics creation program. Of the many graphics tools available, it is one of the most powerful. While it does have a high learning curve, some members of the class already know how to use it, and learning it will be a valuable skill for others. Additionally, it is already installed on all of the computers in the LAC lab and in the Parsons labs.

We decided on a much different approach for our sound creation. We hope to utilize the large amount of free sound effects that can be found on the internet, and only create our own effects to fill in any crucial gaps. If a high enough quality free license sound cannot be found on the internet, the open source Audacity offers an easy way to record and manipulate our own sounds. Since sound will not be a key aspect of our game, it should not require the same amount of developer time or resources.

### **Approaches**

We hope that an incremental, cyclic approach to our software will give us the best results. Every week or so, the product should be at a self-consistent level of completion and polish. The product can then be evaluated and problems and work distributions can be addressed. This also has the advantage that reverting the project to its last successful stage only removes one week of work.

This approach will be extremely helpful if it is possible in creating the artwork. It is likely that the large number of people we must task on creating art content will result in artwork of different styles and complexities. If the artwork can increase in complexity every week, if a certain piece is out of sync with the rest of the project, it will be noticed early and fixed without wasting weeks of effort. Additionally, if the artwork is more or less work than first thought, team members can be reassigned.

The incremental approach also creates advantages for developing the interfaces in parallel to the background code. It helps prevent the divergence between the two groups that might occur if they worked for long stretches without conferring with each other. If a coder thinks that they are supposed to add an unnecessary feature, they are corrected within a week.

Another advantage to this approach is that there is always a workable product within a week's development of the current development. This enables continuous full-build testing, in addition to testing new components as they are added. It also gives each developer a standard framework each week on which to work.

As each group completes their work, they will test their code by integrating it into their own copy of the last build. After they are certain that they didn't cause any conflicts, they will integrate into the codebase that will soon become the next build. By enabling integration to occur as development occurs, we hope to notice conflicts sooner and give more time for resolution.

## Risk and Feasibility Analysis

### Proposed Schedule

We have proposed the following general schedule. It works using a series of builds, each one of which is a working version of the game. After each build, that build is used by every team for their development the next week. As each team completes its work for that build, it is first tested against the old build, and then is then integrated into what will become the new build. After all of the teams code is added successfully, a new build is created so that work can begin again. After each build, there will be a short update during class to keep all of the students on track and maintain their excitement.

Item	Day
Team Component Breakdown	Thurs. 10-1
Team Scheduling	Thurs. 10-1
Master Schedule Completed	Tues. 10-6
Re-balanced Teams	Tues. 10-6
<b>Build 0</b>	<b>Sat. 10-10</b>
Architecture complete, initial working framework	
Group representative meeting	Sat. 10-10
<b>Build 1</b>	<b>Thurs. 10-15</b>
Rough sketches for artwork, interface modifications complete	
Completed Components: Current Assets, Menus	
<b>Build 2</b>	<b>Thurs. 10-29</b>

Most components functioning, consistent decent graphics	
Completed Components: Equations for Resources, Construction Interface, Map	
<b>Build 3</b>	<b>Tues. 11-10</b>
All components functioning, consistent quality graphics	
Completed Components: Logic for Responses, File Access, Hunter/Gatherer, Minigames	
<b>Build 4</b>	<b>Tues. 11-17</b>
All components functioning together as desired ,begin polishing, artwork complete	
Completed Components: Dialogue, Sound	
<b>Build 5</b>	<b>Tues. 11-24</b>
Everything in nearly final form, no features to be added, only bugfixes remain, fix any remaining style issues with artwork	
<b>CODE FREEZE</b>	<b>Tues. 11-24</b>
Everything should be done	Fri. 12-4
Final Packaging Testing	Fri. 12-4 - Tues. 12-8
Everything Due	Tues. 12-8
<b>Work stops</b>	<b>Tues. 12-8</b>

### Proposed Staffing Breakdown

The staff will be broken down into several groups. The group sizes will fluctuate with time, as the needs require. The table gives an initial estimate of staff breakdown for the work leading up to each build.

<b>Build</b>	0	1	2	3	4	5
Art	8	8	8	8	8	4
Testing	1	1	1	1	3	5
Architecture	4	2	0	0	0	0
Mechanics	2	4	3	3	2	3
UI (map, menu, ...)	3	3	2	3	3	3
Hunter/Gatherer	2	2	2	1	1	2
File Access/Dialogue	0	0	2	2	2	0
Minigames	0	0	2	2	1	3
Total	20	20	20	20	20	20

From the table, you can see that there will only ever be six groups active at one time. They are described below.

### *Artwork*

The game will require significant art assets, and as such, a large portion of the class will be devoted to artwork. Their size depends on the skills of the people in the class, which is unknown at this time. However, 8 people for almost the whole project seems a reasonable first guess.

### *Testing*

The game will be tested throughout its lifetime. There will be one or two students whose job is to test the weekly build on people outside of the class to maintain regular feedback on the progress of the game. Additionally, every main group will have a dedicated tester for every build, who will be in charge of making sure that team's work integrates with the old build and has the required functionality. This position will probably rotate within the group. This begins with one person assigned, but ramps up at the end of the project as more extensive testing is conducted.

### *Architecture and Minigames*

This group will create the structure of the project and the object models so that they can be used by the map group, the file access group, and others. After the models are created, this group will work on the minigames, because they would be behind if they joined any other group. This will require 4 people at the beginning of the project, as it is important that quality architecture is created. However, they are not needed after build 1, which is when 2 of them will begin work on minigames.

### *Game Mechanics*

This group will focus at first on creating the current resources tracker. In parallel with the current resources tracker, this group will create the equations that govern resource production and usage. They will then work on the Logic for Response and Dialogue. This group requires 4 people throughout much of the project, in order to write the initial background code and then to balance it at the end of the project.

### *User Interface*

This group will have several large tasks: creating the main screen map, and working on how buildings and tiles appear; creating the building interface that lets the users select buildings and place them on the map; and creating the menus and navigational structure of the game. It requires 3 or 4 people throughout almost the whole project.

### *Hunter/Gatherer*

This group will work on the hunter/gatherer portion of the game. While a much simpler piece to code, this is also a key aspect of the game. This group requires around 2 people to work for almost the whole project.

### *File Access and Dialogue*

This group will work on writing and reading save game files, and creating and voice recording any dialogue. It requires 2 people to create the file routines once the object models are specified, and then 2 people to create the voice acting. This group will also handle sound effects.

## Key Risks and Mitigation Strategies

It is entirely possible that a significant portion of the class's work time will need to be spent creating artwork. The quality of artwork determines to a large degree how seriously the middle school students will take the game. If they think the game looks outdated, they will not give it a chance to prove that it is fun to play. Additionally, artwork is inherently difficult to divide up among a number of people. It is difficult to divide up specific artwork tasks to separate people while still maintaining a consistent style. Being able to work in the same style as someone else is a separate artistic skill from creating the pieces in general.

Hopefully, by working on the artwork incrementally, the team will be able to correct stylistic differences before they become too serious. The other advantage of incremental artwork is that the game is always self-consistent with itself. It enables re-balancing of the work if some people find that they cannot create the assets as quickly as they originally thought. Additionally, after each weekly build, the group sizes can be rebalanced if there is drastically too much or too little work.

Additionally, there might not be enough skilled computer artists in the class to take on such an ambitious project. To solve this, the artwork quality could be scaled back across the board at the beginning of the project, after a survey of the class skills has been taken.

Whenever such a large group is tasked on a single project, teamwork issues are impossible to avoid. By splitting the group up early, we plan to eliminate some of the really large group dynamics. By having large, 10 minute overview meetings during class every 10 days, we hope to keep the large group aware of what the other groups have accomplished, and keep each individual excited about the project.

There is a large piece of groundwork that needs to be accomplished at the beginning of this project before most of the groups can start their work. The first week of the project has been set aside to let each team look at what they need to accomplish throughout the whole project, while enabling a separate team to lay down the code foundation. Without a good idea of the object models and function interactions, the rest of the project will work very inefficiently.

Having sufficient testing throughout the process is an issue the group must be aware of. Throughout the project, testing is a one or two person job. However, at the end of development, the product must be rigorously tested. By including extra slack time at the end of the schedule, problems that are discovered through this testing can be effectively dealt with, leading to a successful product launch within schedule.