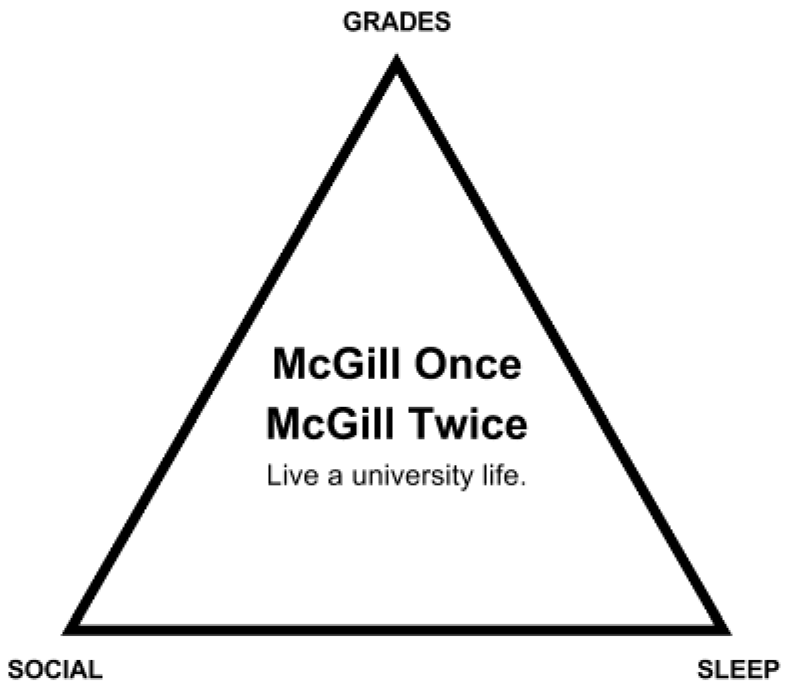


PRELIMINARY DESIGN DOCUMENT

COMP 361: Software Engineering Project
School of Computer Science
McGill University, Montreal, QC

Monday, October 26th, 2015

Othniel Cundangan	**Aliaa Moharram**	**Tristan Struthers**	**Ching-Chia (Russell) Wang**	**Elizabeth Wu**
260575092	260572705	260568567	260517682	260571586

PROJECT OVERVIEW

Team Members

Name	McGill ID	Team Role
Othniel Cundangan	260575092	Scrum master/Graphics/Programmer
Aliaa Moharram	260572705	General C# Programmer
Tristan Struthers	260568567	General C# Programmer
Ching-Chia (Russell) Wang	260517682	General C#/Script Programmer
Elizabeth Wu	260571586	Graphics/Server Programmer

Client Information

Our project ideally targets an audience of older teens and/or university students. The client is a high school student curious about university life, or a current university student who would like to experience different social and academic settings within McGill.

Project Overview

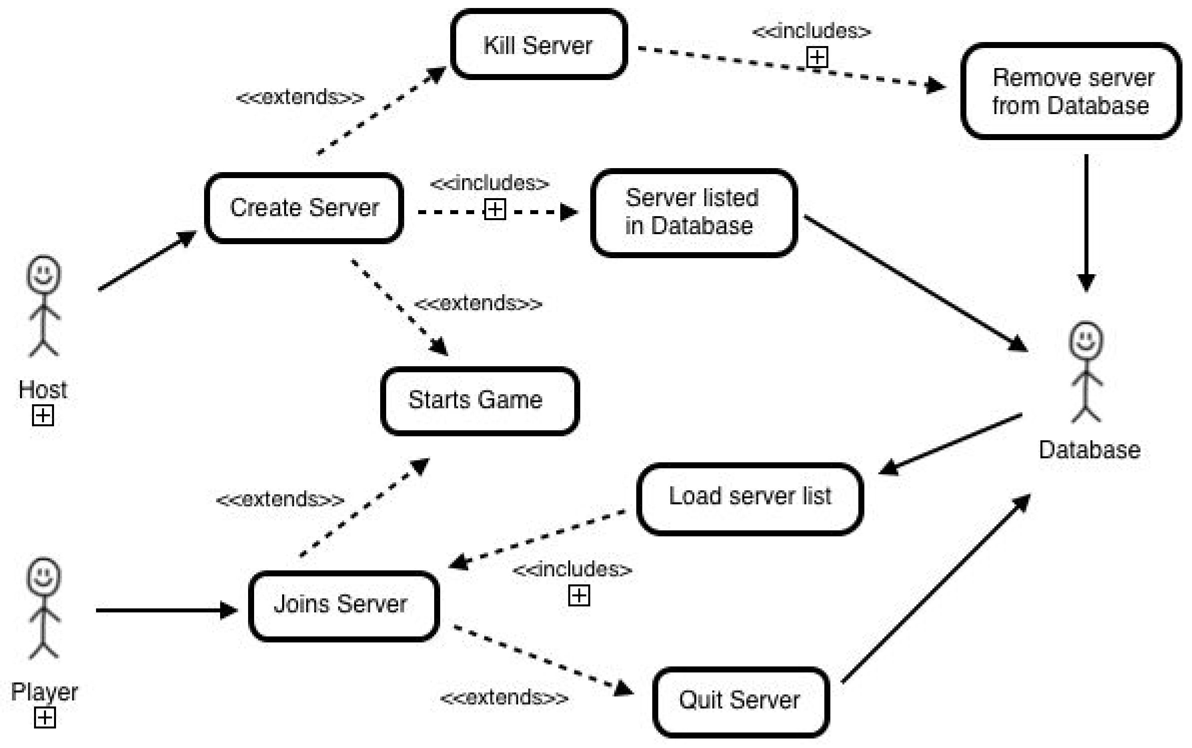
The stakeholder would like to build an evolving open world game with attendance at optional side adventures. The user should be able to pick an avatar from a selection of at least 2 options. Once the world is created the avatar is placed randomly somewhere and the avatar must be able to survive and prosper in this world.

Our end product will simulate life at McGill by allowing the player to participate in activities rooted in McGill culture. The player will have 3 bars that determine their survival: Sleep, Grades, and Social Life. They interact with the world in order to keep at least 2 out of the 3 bars somewhat filled. Interactions will include items similar to attending class, visiting friends, and sleeping in Trottier. Expulsion (death) occurs if 2/3 bars become empty. To increase the replayability of the game, each session's world will be different. Though the campus will remain static, the structure of everything surrounding it will be procedurally generated. This includes the ghetto, overall layout of the city, and the location of your (NPC or RL) friends. The game will have network capabilities so you and your friends can live outrageous lives when you're not out living your own. The game will be comprised of 3 main modes: Free roam, where a player can explore the surrounding area unrestricted. The second mode is comprised of a series of minigames, where the player is subject to certain constraints and is challenged to complete an objective. The third mode will be a house customization mode, where players will be able to see their living space and place certain objects as they see fit.

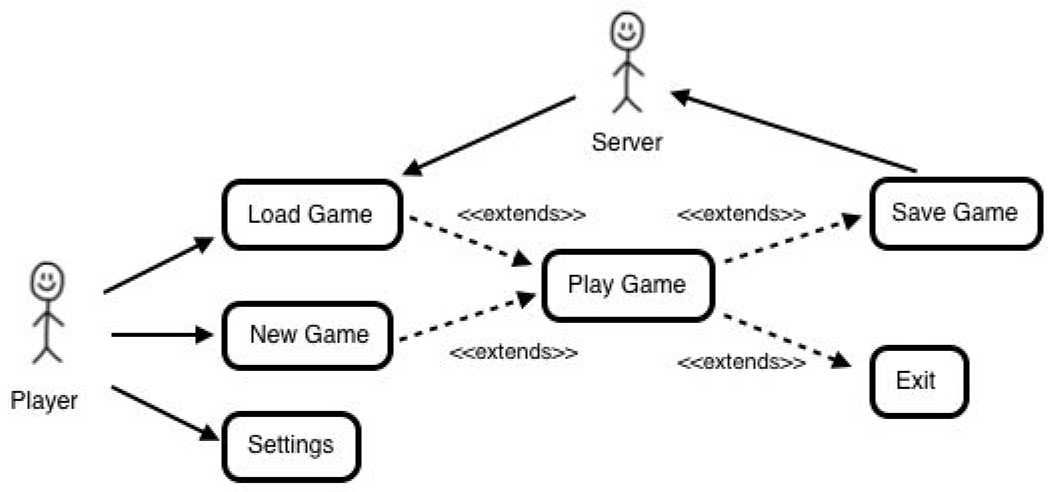
The baseline for the project is a mobile application created by EA mobile called "Surviving High School." Our project will resemble the gameplay style of text based interaction with NPCs with decisions that directly correlate to the outcome of the story in game. The software will be created using Unity, with scripts written in C#.

SYSTEM DESCRIPTION

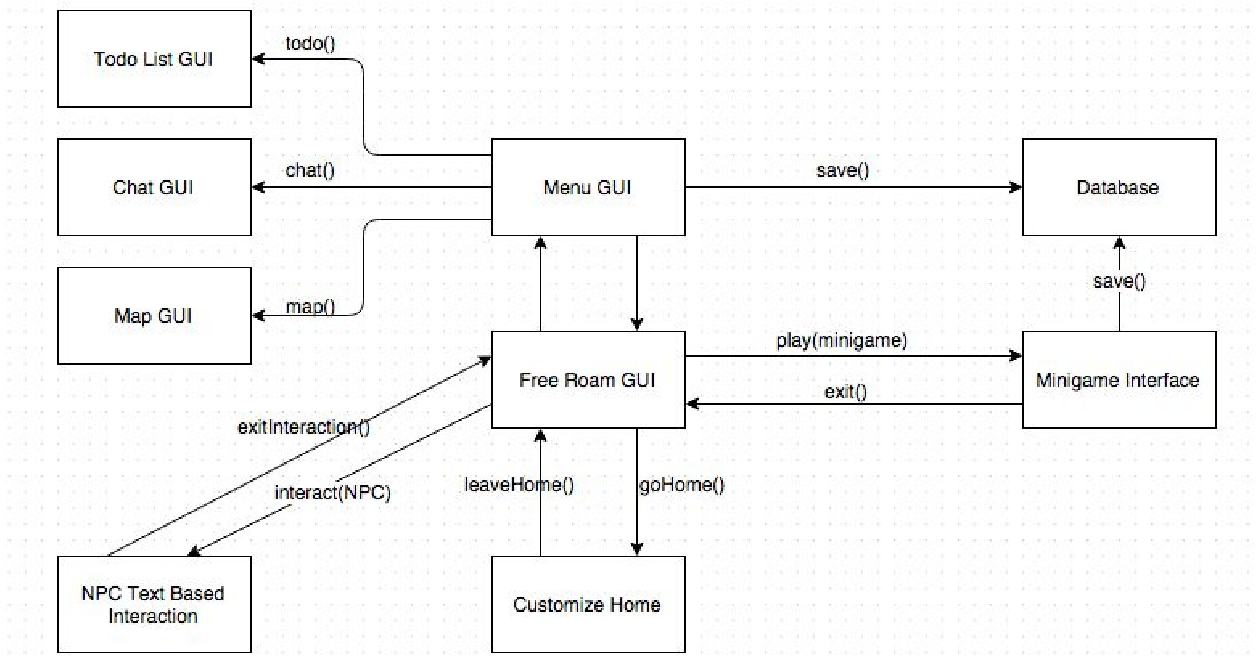
Multiplayer Initial Setup Use Case



Single Player Main Screen Use Case



Gameplay Block Diagram



Technology Selection

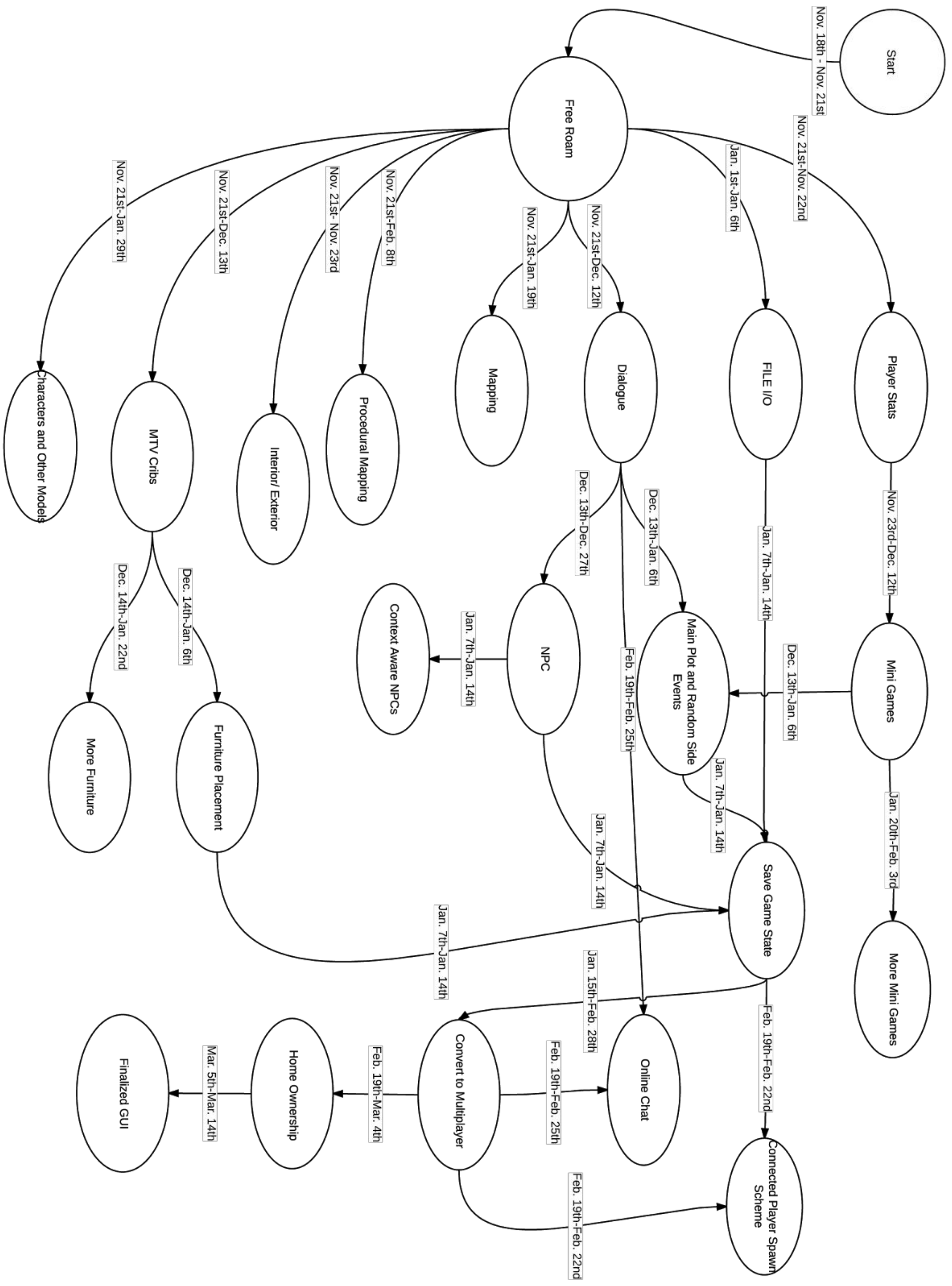
The game will be built upon the personal edition of Unity's engine and make use of Unity's provided solution for multiplayer connectivity called uNet. The structure of the game requires that there be at least a server in order for there to be a game running. The client may choose to be just a host, just a player, or both a player and a host in which case, the player will be playing within the game they are hosting. In order to save and load the state of the game, the program will require local write permissions to a designated folder on the hard disk and to keep things simple, the Json.NET API will be used for serializing and deserializing objects in order to write to and read from a file. To build the 3D assets such as characters, maps, and props, the open source application Blender3D will be used in conjunction with an image editing program.

PHASED BREAKDOWN

ID	Task	Assigned Person	Duration (Days)
1A	**Free-Roam Functionality** Player can move around in the environment.	**Othniel**	**3**
2B	**Interior/Exterior Transitions** Allow the player to “enter” buildings. Create a teleporting object that links one world position to another.	**Aliaa**	**3**
3C	**Player Stats Functionality** Give the player stats for social, sleep, and grades. Create placeholder GUI to show the stats. Implement player death and respawn.	**Tristan**	**3**
4D	**Minigame Model** Use UML diagrams to design a generic class from which different types of minigames can be derived from. Also make the functionality for one minigame for each different stat.	**Russell**	**15**
5E	**File I/O** Create a system that allows quick file writing/reading from a location on the hard drive.	**Liz**	**6**
6F	**Save Game State** Allow saving the game’s state. Serialize all dynamic objects as strings and write to a file kept on the machine. Possibly use JSON.net API to make this simpler. Allow a game to start from a saved state. This includes player state and their inventory.	**Othniel**	**8**
7G	**Player Inventory and MTV Cribs Catalogue** Sketch the GUI for the catalogue and inventory. Implement the inventory and catalogue with placeholder GUI so they interact. ie. a purchase from the catalogue places the object in the player’s inventory.	**Aliaa**	**14**
8H	**MTV Cribs Home Rearranging** Allow players to place items in their crib. Changes to the environment should be sent to the server and relayed to other players.	**Tristan**	**13**
9I	**Dialogue System** Sketch the GUI for a dialogue system. The system will be used to tell the client of what’s happening. Ex. the client interacts with an NPC, the NPC says something through the system such as “yo it’s a lit party on aylmer.” Ex2. the client receives a notification from an NPC character inviting them to a lit party. Opening the notification initiates the dialogue system and tells them about the party.	**Russell**	**16**

10J	**Main Plot and Random Side Events** The player can follow the main plot. At random intervals, the player will also be invited to side events and minigames.	**Liz**	**16**
11K	**NPCs** Create NPCs as automated player avatars. Make them interactable.	**Othniel**	**10**
12L	**Online Text Chatting** Design GUI to show a global text chat. This is different from the dialogue system in that it's always shown.	**Aliaa**	**5**
13M	**Map of McGill Campus** Create the McGill Campus environment. The map should have a decent amount of buildings and event sections to sustain a lengthy play time.	**Everyone**	**60**
14N	**Finalize Spawn Scheme** Decide on how the player spawns on the map when connecting to a server.	**Russell**	**4**
15O	**Finalize GUI Elements** Replace all placeholder GUI spaces with functional graphics.	**Liz**	**8**
16P	**Context-Aware NPC Interactions** NPCs tell you about events going on.	**Othniel**	**6**
17Q	**More Minigames** See title.	**Everyone**	**15**
18R	**Home Ownership** Create restrictions based on the player ID for placing furniture.	**Tristan**	**8**
19S	**Larger Selection of Furniture** Expand the MTV Cribs catalogue.	**Everyone**	**10**
20T	**Characters and Other Models** Create models for player avatars and other props.	**Everyone**	**70**
21U	**Procedural Mapping** Generate parts of the map based on random seeds.	**Everyone**	**80**

DATED ACTIVITY GRAPH



MILESTONES

Milestone	Deliverables	Completion Date
Free-Roam Functioning Players are able to walk around freely in the environment and approach objects that will take them to designated minigame locations. The minigames themselves won't be functioning, however the location will be available.	A build of the game where the client may roam free in the game's environment.	**Nov. 20**
Minigames Functioning Players are able to trigger minigames by going to specified locations. Minigames will increase the corresponding stat when completed successfully.	Game build. UML class diagrams for objects representing players, scene transitions, and minigames.	**Dec. 12**
NPCs Functioning NPC characters wander the environment and are interactable.	Game build. UML class diagrams.	**Dec. 30**
MTV Cribs Functioning Players can enter designated buildings and add furniture/customize them after making purchases from a catalogue.	Game build. UML class diagrams for objects representing furniture, player inventory, and the catalogue. 3D models of some placeable furniture.	**Jan. 10**
Plots Functioning A main plot that the client can follow to the end has been created. Plots consist of minigames and a sensible storyline. The client receives notifications at random intervals inviting them to a minigame or side plot.	Game build. Plot storyboards.	**Jan. 10**
Game State Saveable The state of a game can be saved and loaded. This includes all the players' states and customizeable homes.	Game build. UML class diagrams.	**Jan. 18**
Random City Generation The game's environment will be generated based on random seeds. A game's environment will only be generated once, then saved. Later sessions of the same game use the saved environment.	Game build. Outline of generation procedure.	**Feb. 12**
Conversion to Multiplayer Refactor all the classes to support communication with a server.	Game build.	**Feb. 22**
Final Product Finalize everything and build the product.	Final build.	**Mar. 18**

RESOURCE REQUIREMENTS

Item ID	Description	Unit Cost	Quantity (Units)	Subtotal	Dependencies (IDs)
A	**Computer/Laptop**	**Covered**	**5**	**0**	**--**
B	**Unity**	**0**	**5**	**0**	**A**
C	**Blender3D**	**0**	**5**	**0**	**A**
D	**GIMP/Paint.NET**	**0**	**5**	**0**	**A**
E	**GitHub private repo**	**0**	**1**	**0**	**A**
F	**Trottier group study area**	**0**	**1**	**0**	**--**
G	**Working day**	**1 day**	**86**	**86 days**	**ABCD**

COST ESTIMATE

This project requires no financial costs. As indicated in our technology section, we will be using Unity's free personal edition. Our multiplayer interface will be designed around one of the clients being the "host" removing the cost of a dedicated server. We will be making use of Trottier's group study areas (free access to computer science students) as well as our personal computers to write code. A GitHub private repository (free student promo) will be responsible for version controlling.

The real cost of development will be time. As full time students, the team will be spending valuable time developing and testing the software during the active semester. We will budget our time appropriately using the software development techniques and patterns described in the following section. The team will actively attempt to adhere to the milestones/deliverables table and make alterations as necessary.

COCOMO Estimate

Class: Organic
LOC: 18,000
Due: 4 months
Effort: $2.4 * 18^{1.05} = 50$
Months: $2.5 * 50^{0.38} = $ 11 months/person

	Effort	Months	Schedule	People
Req./Spec.	30%	3.3	1.2	2.75
Design/Code	40%	4.4	1.6	2.75
Integrate/Test	30%	3.3	1.2	2.75

ENGINEERING METHODS & TEAM WORK ENVIRONMENT

Software Process Model Selection: Prototype

Instead of defining the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements. By using the prototype method, quicker user feedback is available leading to better solutions and confusing or missing functionality can be identified easily

Programming Development Method: Iterative

An iterative development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements. We work iteratively by creating a rough product or product piece in one iteration, then review it and improve it in the next iteration and so on until it's finished.

Team Organization: Scrum - Agile team organization.

Our team is built as a collection of peer programmers. There is a "Scrum Master" who is responsible for facilitating the team, obtaining resources for it, and protecting it from problems. Other team members act as generalist programmers and are responsible for the creation of the program. There is no librarian, instead we'll use an integration station for testing.

Process Assessment Rules:

Feature testing and ease of use are the main methods of this project. The project's goal is maximize client and stakeholders' enjoyment. Aesthetics is a main component of the marketability of the game, with speed and memory usage coming close after. Peer play testing and recording of impressions will be used to predict our target audience's reaction of the game.

Change/revision control methods:

Project code will be uploaded to Github. Version control allows the team to access and view previous versions of the code. Any requirement changes will result in a new branch, but past versions will be saved should the team need to revisit or start over. The team will make use of best practice git techniques including branching and merging to collaborate on code and maintain a history of changes made to the source code.

RISK IDENTIFICATION & IMPACT ESTIMATION

Risk - Multiplayer Capabilities: Multiplayer may be too complex or difficult to implement within the given time frame. Even if we set up multiplayer function it could be rudimentary and faulty.

Impact: Possibly an additional month before completion, plus client disatisfaction.
Solution: First build the game to work in a single player campaign mode, then in a later iteration, convert all the single player functionality to support multiplayer.

Risk - Graphics Support: Ideally we want to recreate the McGill campus, but graphics of the game depend on Unity's constraints and the team's graphics experience. This is a major risk because the main appeal of the game is the fact that the setting takes place on a well-known University. The more realistic the campus, the more attractive the game is to clients.

Impact: Could take several more several weeks to create the map.
Solution: If the graphics we want aren't supported, or implementable, we should avoid this risk, scale back and simplify the graphics

Risk - Bad Reaction From Target Audience: Although the team believes that this will be an interesting product to the target audience, it's possible the team's vision is not consistent with the demand of the public. The concept of the game may not appeal to the incoming college students.

Impact: The finished product will not be marketable.
Solution: Create time before the final release for beta testing, where we have opportunities to expose a version of our project that is representative of our final piece to a sampling of the target audience. We can record the reactions and impressions of such a population and make necessary changes to our product in hopes that we can preemptively improve our project in the eyes of our target audience.

Risk - Procedural Map Too Complex To Implement: Generating a map procedurally from seed values may prove to hard to implement. There are very complex techniques of generation, and then simpler methods that are less random.

Impact: Every game may potentially have the same static map. Additional time may also be required in order to finish the product.
Solution: If the procedural technique in use cannot be implemented within the timeframe, scale back the procedure to a simpler method. For example, instead of using buildings, streets, and intersections as building blocks for the city, use templates for whole city blocks to produce less random, but easier to manage cities.