

UVic Breakout

Requirements Specification and Conceptual Design

Ability Software

Brayden Arthur	Project Lead
Abhi Jagdev	Application Design
Gabriel Silvarredonda	User Interface
Isaac Streight	Toolsmith and Analyst

Table of Contents

Glossary of Terms	4
Project Summary	5
Objectives	5
Vision	6
Role-Playing Game and States	6
Location Based	6
Characters and Items	6
User Interaction	7
Global UI Elements	7
(1) Buttons	7
(2) Scrollbars	7
(3) Titles	8
(4) Pop Ups	8
Screen Types	9
Screen Type 1 – Landing	10
(1) Navigation Buttons	10
Screen Type 2 – Zone	12
(1) Title	12
(2) Map Button	13
(3) Image	13
(4) Description	14
(5) Actions	14
Screen Type 3 – Inventory	16
(1) Title	16
(2) Description	16
(3) Actions	17
(4) Item List	17
Screen Type 4 – Map	19
(1) Map	19
(2) Back Button	20
Screen Type 5 – Text-Only	21
(1) Title	21
(2) Text	21
(3) Button	22
Environment	23
Story Progression	25
Story Completion	26
Side Quests	27
Management Plan	27

Minimal Systems Requirements	27
Implementation	28
Application Architecture	30
UML Diagrams	30
Sequence Diagram	30
Use Case Diagrams	31
Story Board Diagram #0	32
Story Board Diagram #1	33
Story Board Diagram #2	34
Story Board Diagram #3	35
NPC and Item Interactions	36
Interaction Testing	37
Code Outline	37
Table Structure	38
Ethics	39
Data Saving & Storage	40
Saving	40
Automatic Saving	40
Manual Saving	40
Testing	41
Storage	41
Location	42
External	42
Internal	42
Ethics	43
GPS	44
GPS Features	44
Determine Current Geolocation	44
Location Manager	44
Location Provider	45
Proximity Alert	45
Security	45
Prompt the User to Enable GPS	46
Ethics	47
Testing	47
Android Testing Support Library	48
Monkey Tool	48
Monkeyrunner	48
Document Summary	49

Glossary of Terms

RPG - Role-Playing Game, generally has an adventure type story and allows the player to make decisions to influence the story or their own character.

GPS - Global Positioning System, a way to track the player's location within the game.

NPC - Non-Player Character, a character within the game that the player can interact with, but cannot make decisions for.


Character - A synonym for NPC in the context of this document.

UVic - University of Victoria, the setting for UVic Breakout.

Project Summary

This document is in response to Almost Games' RFP for a team to develop a GPS Based Text-Adventure RPG video game. The requirements and constraints of the game are described, as well as desired aspects of the final product. Project objectives are outlined along with the user-product interaction.

Objectives

The proposal requests a GPS Based Text-adventure RPG Video Game. **The game must be completed within a 3 month period.** 

The game will include RPG-like elements mirroring game styles of 80's text adventure games.

The game will incorporate a modern element, GPS and smartphones, into the old genre. The game will be playable as an application on a smartphone. The in game map, of which the player will traverse, is a replication of the UVic campus.

In order to visit locations in the game and progress through the story, the player must also physically visit the in game locations in real life. The GPS on the smartphone will detect when the player is in the right location and progress the game.

Riddles based on landmarks may be incorporated into the game to incentivize the physical aspect. Though, users will generally interact with the game using the traditional text based game approach. It is intended that the in-game setting of UVic will have some themes within certain areas, game story will be designed around those themes.

The story of the game is intended to be satiric, heartwarming, and ridiculous. A variety of stories will be present in the game, a main story as well as smaller side stories.

Vision

Role-Playing Game and States

The game will incorporate traditional elements of RPG, such as quests, items, secrets, and levels. The game will run in a state to state manner, a player's current state consists of their items, level, prior interactions, and their point in the story, from the current state the next state the player can enter is determine from their decisions and current state elements. The number of states the game includes will be negotiated, though it should so much that through one typical playthrough less than 50% of the state elements are used. Inclusion of puzzles and riddles is encouraged though difficulty should remain easy to not induce frustration in the player. Hints should be readily available to a player. **No player should get stuck for an elongated period of time.** Emergent gameplay and multiple ways to progress through a game element is a stretch goal.



Location Based

The game will make interesting use of the GPS built into smartphones. The in game map will be a replica of the floor plan and environment of areas at UVic. The player's location in real life will be synced to the player's location in the game. The GPS on the phone detects when the user in is in a specific location and triggers game elements accordingly. The in game map will load a map for each location of interest the player walks into. The outside UVic campus will have its own map as well as buildings of interest.

Characters and Items

The game contains a variety of characters, items, and character skills. Characters will populate the game world at a rate of about 1-2 per area. The characters will offer the player insight, advice, items and other game-related elements. Some characters will attempt to harm the player. Items are available to the player from character interaction and finding them in the map. Randomization and luck in finding items will be used sparingly. Items include consumables and weapons. Consumables are used by the player and affect player status and other game elements. Weapons are mostly aesthetic since the game does not feature an active combat system.



User Interaction

Global UI Elements

Some UI elements are consistent throughout the system. They are:

1. Buttons
2. Scrollbars
3. Titles
4. Pop Ups

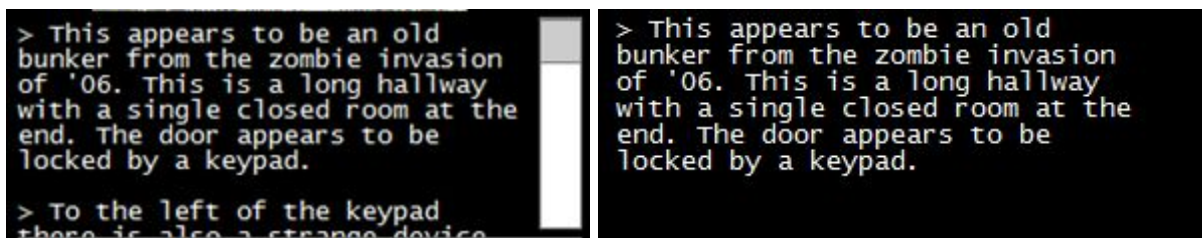
(1) Buttons

Buttons are clickable UI elements which the user can tap. They perform some game action or show a pop up.



(2) Scrollbars

The user can use the scrollbar to see additional options or text in an interface component, or tap and drag up or down anywhere to the left of the scrollbar to achieve the same effect. If there are no additional options or text to be displayed in the component, the scrollbar will be hidden and disabled, and touch interactions will be similarly disabled.



(3) Titles

Titles for pages may be static or dynamic. Dynamic titles will be used for Zone type screens. These titles will be generated from the name of the Region and Zone the player is in. Additional information on dynamic titles for zones can be found under the Screen Type 2 – Zone section.

(4) Pop Ups

Pop ups will present the user with a maximum of 256 characters of plain text and up to two small buttons. They will always have at least one button that closes the pop up or a time delay before the pop up disappears. The pop ups may include a warning indicator.



Screen Types

There are five possible screen types that will be presented to the user. They are:

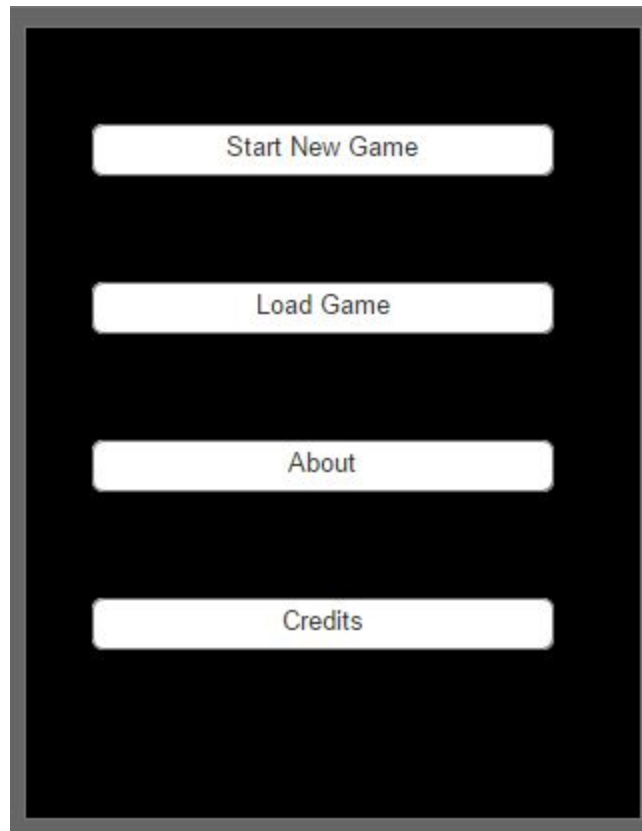
1. Landing
2. Zone
3. Inventory
4. Map
5. Text-Only

Each screen may be divided into multiple components. Each component may contain some number of Global UI Elements, coloured dividers to separate components, images, and/or plain text.

The following pages outline each possible screen type. Note that for each screen type, multiple variations in text and images are possible, but only one example of each is presented in this document.

Screen Type 1 – Landing

The Landing screen only has one component, which consists of four buttons.



(1) Navigation Buttons

There are four buttons on the Landing screen.

Possible Interactions

The Start New Game button will begin a new game for the user, overwriting their previous save file. A confirmation box will pop up asking the user: "This will delete any previous save file. Continue?" which will allow the user to tap either "Continue" and begin a new game, or cancel which will close the pop up and take no further action.



The Load Game button will load the saved game file and take the user to the respective screen for their current GPS location. If GPS location cannot be verified, a pop up will inform the user: "GPS location cannot be verified, do you have location services turned on?". The pop-up will have an OK button which will close the pop up and take no further action.



The About button will take the user to the About page, a Text-Only type page which will display information about the game. Specifically, it will have the Introduction from the UVic Breakout User Manual.

The Credits button will take the user to the Credits page, a Text-Only type page which will display the names of the creators, developers, funders, and additional supporting individuals and organizations that contributed to UVic Breakout.

Screen Type 2 – Zone

The Zone screen is divided into five components. There are many different Zone screens, each with their own title, description, image, and possible actions. Below is an example of one of these screens.



(1) Title

Each zone will have its own title to remind the player where they are. This title will be split into two parts, the first will tell the player what game zone they are in, and the second will tell them what part of that zone they are in.

Zombie Bunker
Entrance

(2) Map Button

The map button will be present in all Zone type screens.

Possible Interactions

Clicking the map button will take the user to the map screen.

(3) Image

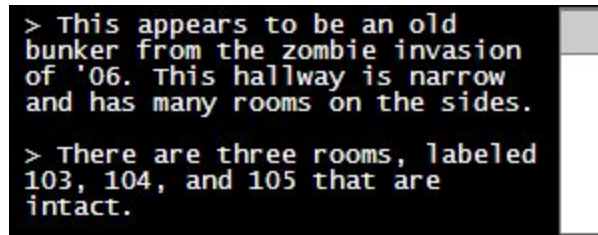


Each zone the player is in will have its own image, either an edited image of the real UVic at that location, or a picture of the UVic map with their position marked on it. The edited images will mostly be a picture of the building or area the player is currently in, and may have some edits to go with the theme for that area. The images will also be pixelated to fit in with the style of the game in general.



(4) Description

Below the image will be a text description of their situation. This might include details such as their current status, what they see, and where they could go.



```
> This appears to be an old
bunker from the zombie invasion
of '06. This hallway is narrow
and has many rooms on the sides.

> There are three rooms, labeled
103, 104, and 105 that are
intact.
```

Possible Interactions

This component has a scrollbar. Interaction with the scrollbar is described in the Global UI Elements section.

(5) Actions

Finally, at the bottom of the screen will be a list of options the player can perform. This list will have a scrollbar if there are many possible options, but it will usually be limited to 3 or 4. This will also include the 'Inventory' button, which will always be available for the player to view their collected items.



The actions menu may also have different action types based on the interaction the player need to perform, an example of entering a code for a keypad is shown below:



Possible Interactions

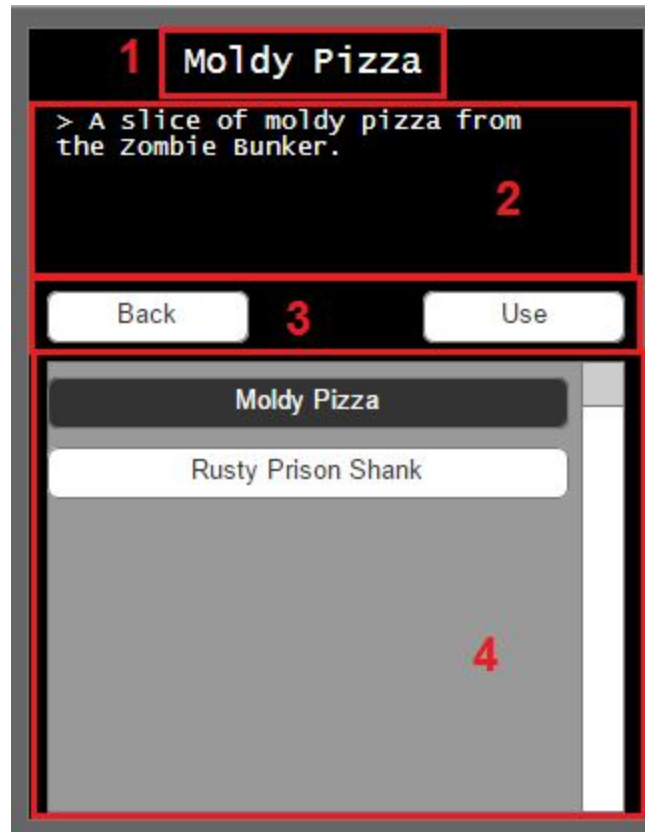
The Inventory button will take the user to the inventory screen.

The other buttons will take the user to a new 'zone' within the game, generally these will look similar, but with different images and actions.

The text box will open an onscreen keyboard and allow the user to enter characters. This text box will flash a red outline for one second if the user taps the submit button before entering anything, or if they enter the wrong code.

Screen Type 3 – Inventory

The inventory screen itself will work much like a larger actions menu, with a smaller description, no image, and no map link. It will have a back button and use button. The top item in the list will always be selected by default when the user enters this screen. The inventory screen is separated in four components.



(1) Title

The title will be the name of the currently selected item.

(2) Description

The description of the currently selected item. Does not have a scrollbar and is limited to 256 characters.


```
> A slice of moldy pizza from  
the Zombie Bunker.
```

(3) Actions

A Back button and a Use button, interactions are described below.



Possible Interactions

Tapping the Use button with an item selected will have the system check if that item would have an effect in the user's current zone, if so, it takes the user to another zone, based on where the user was and what item the user used.

Tapping the Back button will take the user back to the previous screen.

(4) Item List

A list of the items the player has, displayed as buttons. This list is dynamic and will be generated each time the player enters the inventory screen. The selected item will be indicated by a black background.



Possible Interactions

This component has a scrollbar. Interaction with the scrollbar is described in the Global UI Elements section.

Tapping an item in the inventory list will change the description to that of the item.

Screen Type 4 – Map

The map screen will be a single screen that can be viewed from any zone in the game. It will show a large map of the UVic campus and have a red marker indicating the current location of the user, based on GPS data. The map screen is separated in two components.



(1) Map

The map will be a static image. A red marker will be overlaid on the map at the estimated position of the player based on their GPS location. If GPS location is unavailable, no marker will be shown.

Possible Interactions

Tapping the map will display a pop up that says “Move around UVic with GPS enabled to change locations in the game.” It will have an “Ok” button to close the pop up.

(2) Back Button

The Back button will take the user back to the previous screen.

Possible Interactions

Tapping the back button will take the user back to the previous screen.

Screen Type 5 – Text-Only

This screen type will be used for the About, Credits, and Game Completion screens. It consists of three components.



(1) Title

The title will be static. Different Game Completion screen will have different titles.

(2) Text

This will be a large text box able to support 4,000 characters.

Possible Interactions

This component has a scrollbar. Interaction with the scrollbar is described in the Global UI Elements section.

(3) Button

The back button will take the user back to the Landing screen.

Possible Interactions

The Back button will take the user back to the Landing screen.

Environment

The environment of UVic GPS(Global Positioning System) enabled text-based RPG(Role Playing Game) is based around the campus of the University of Victoria. While the environment will not be identical to the University of Victoria campus, it will have similar aspects to the University of Victoria environment.

The environment will be segregated into large areas based on the layout of the University of Victoria. As an example, the location close to the Engineering Lab Wing (ELW) and the Engineering and Computer Science (ECS) building would be centred around technology and invention.

The in-game environment will be bounded by physical, GPS related boundaries. If a player physically walks outside of the boundaries, there will be in-game consequences. These in-game consequences may include injury, capture, or even death.



The game provides an in-built auto save function to be saved every time a new location is archived. The archived versions are then available to the players to visit and pick of the game where they left. The archives also allow the player to replay the different unlocked stages to try perhaps a different path to success like in SpaceShip Quest (check the story line).



Story Progression

The game will be primarily narrative and the story will be a major emphasis on the story. However, the game will be designed to be short. This means the story must have appealing elements beyond pure character building. The story of the game is not explicitly defined, yet it will have emphasis towards relating with university students, primarily to those at the University of Victoria. The story will be satirical, exciting, and uplifting, with areas that instill fear and a sense of ridiculousness.

The conflicts of the game will primarily be in the form of puzzles. This replaces the need of an active combat system, and staying more true to the text-based RPG genre. However, there will still be combat and conflicts, but within the parameters of the text-based RPG genre. The puzzles will vary in difficulty and style. All styles of puzzles will include some, and varying, levels of physical world interaction. One style will be similar to solving a maze and another will be of a lock-and-key style. In the lock-and-key style of puzzle, the player will be presented with a lock, for a door, chest, etc., and must find the 'key', or code, in a different location in the physical world. As an example of the lock-and-key style of puzzle, the player may be presented with a locked location in-game while on the first floor of the building X in the physical world and the code to unlock the location will be the address of building Y.



The game will include non-player characters (NPC). These NPCs will interact with the player, offering items or quests, providing conflicts, or acting as a shop. Some NPCs will be integral to the game, some to a specific storyline, and some will be completely optional. NPCs will add to the emotional aspect of the game, with witty one-liners or offering heartwarming feelings.

As one of the RPG elements of the game, the game will include items and an inventory system. Items will be collected as rewards, through NPCs, or by exploration. Like NPCs, items may be integral to a specific storyline, aid the player, or even hinder the player.

Story Completion

The game may be completed in multiple ways.

Business	The business completion strategy involves earning enough currency, via quests or exploration, to open an in-game shop and set up a trade route between the player owned shop and NPC shops. This ending offers the player character a reason to remain the game environment and able to survive comfortably.
Completionist	The completionist storyline involves the character completing all the areas of the in-game world. This involves completing all puzzles in each location. This offers the player character a sense of mastering and ownership over the in-game world and thus a reason to remain.
Cultural	The cultural completion method involves resolving all major conflicts peacefully. This ending unites the locations of the in-game world and the NPCs revere the player character as a hero, and thus offers a reason for the player character to remain.
Engineering	The engineering-style completion is accomplished by accumulating in-game items to build a spaceship to escape the in-game world. The parts may be found via exploration, interaction with NPCs, or via a crafting. This ending offers the player an escape from the in-game world, back to the player character's original home.
Philosopher	The philosophical completion strategy involves the the player character finding inner peace. This is accomplished by completing one stage of each of the other completion strategies. This ending offers the player character a sense acceptance of their place in the in-game world, and thus a reason to remain.
Science	The cultural completion method involves the categorization of species of the in-game world. The species may be discovered via exploration or interaction with NPCs. The game ends upon discovery of all species. This ending offers the player character a sense of order and accomplishment within the in-game world and thus a reason to remain.

Side Quests

While the story is the main focus of the game, it will also offer side quests. This offers additional stories and activities to the game. These side quests will enforce the ideas of exploration, puzzle completion, and conflict. These side quests will involve more and different chance than the main quests, offering additional replay value.

Management Plan

Minimal Systems Requirements


The product is a video game for mobile platforms, it will be released on Android, initially and an iOS version release in the future. The game will be available on the mobile device's respective app stores.



The game will be **available** only on the current version of Android Lollipop version 5.1.X and it will not be backwards compatible. It is entirely possible that the application will work on previous versions of software but the developer bears no responsibility for any damage to hardware or software if the mentioned specific requirements are not met. Along with that, It is recommend that the Android software support for OpenGL is also present on the hardware, although this is a included in most Android platforms the developers take no responsibility in ensuring and checking that the OpenGL is working with all its dependencies on the current device.

The hardware requirements for the project to operate smoothly and seamlessly are listed below:

- Android Smartphone with the following specifications:
 - Operating System: Android 4.0 "Ice Cream Sandwich" or later
 - System on chip: Qualcomm Snapdragon S4 MSM8960
 - CPU: 1.5 GHz dual-core Krait
 - GPU: Adreno 225
 - Memory: 2 GB

- Storage: 16 GB (minimum) 
- GPS Sensor (working & tested)

** For any further clarification on compatibility please visit the Google's Android Compatibility Definition for Android "Ice Cream Sandwich" 4.0.X (<http://static.googleusercontent.com/media/source.android.com/en//compatibility/android-4.0-cdd.pdf>)

Implementation

The implementation of the game will be done entirely in Android Studio. The team will be using a combination of different platforms among themselves to coordinate the development.

Android Studio is the official IDE for Android application development, based on IntelliJ IDEA. Android Studio uses Java for functionality, and XML for GUI and UI.

The Android API for GPS sensors is also used for retrieving a user's current location on the smartphone to progress the game. Most Android devices allow applications to determine the current geolocation. This can be done via a GPS module, via cell tower triangulation or via wifi networks.

Android contains the `android.location` package which provides the API to determine the current geo position. The `LocationManager` class provides access to the Android location service. This services allows to access location providers, to register location update listeners and proximity alerts and more. The `LocationProvider` class is the superclass of the different location providers which deliver the information about the current location. This information is stored in the `Location` class.

The Android device might have several `LocationProviders` available and you can select which one you want to use. In most cases you have the following `LocationProviders` available.

LocationProvider	Description
network	Uses the mobile network or WI-Fi to determine the best location. Might have a higher precision in closed rooms than GPS.
gps	Use the GPS receiver in the Android device to determine the best location via satellites. Usually better precision than network.
passive	Allows to participate in location of updates of other components to save energy

While GPS will be used extensively in the game, precise locations will not be used. The GPS element of the game will guide the player to a general area (ie, Library area) where the player can interact with the location environment, NPCs, and items. This will reduce errors due to GPS precision.

The Google Maps API will also be used for pointing the locations of the users and characters within the map. Following is a simple example of Google Maps API in use setting a location point on Google Maps

```
<resources>
<string name="google_maps_key_instructions" templateMergeStrategy="replace">!--
TODO: Before you run your application, you need a Google Maps API key.

To get one, follow this Link, follow the directions and press "Create" at the end:

https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX%3Bteamtreehouse.com.iamhere

You can also add your credentials to an existing key, using this line:
XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:teamtreehouse.com.iamhere

Once you have your key (it starts with "AIza"), replace the "google_maps_key"
string in this file.
--></string>

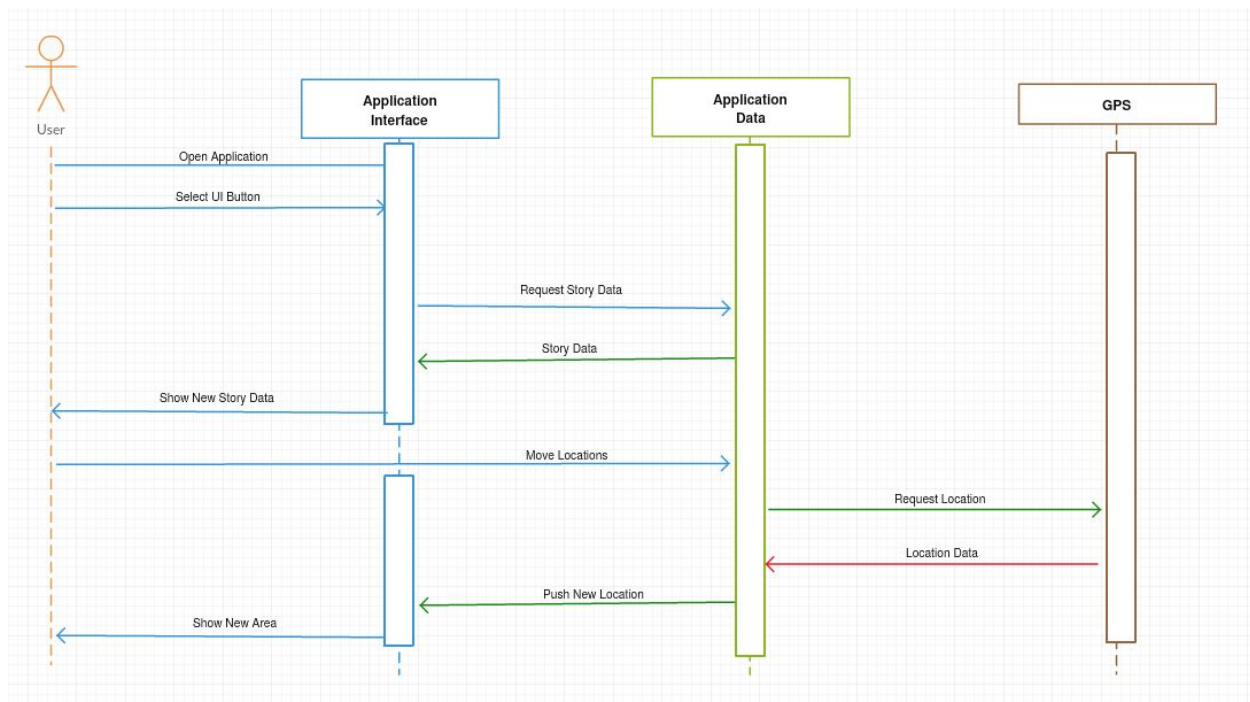
<string name="google_maps_key" templateMergeStrategy="preserve">YOUR_KEY_HERE</string>
</resources>
```

** Further information about Android Studio can be accessed via their website (<http://developer.android.com/tools/studio/index.html>)

Application Architecture

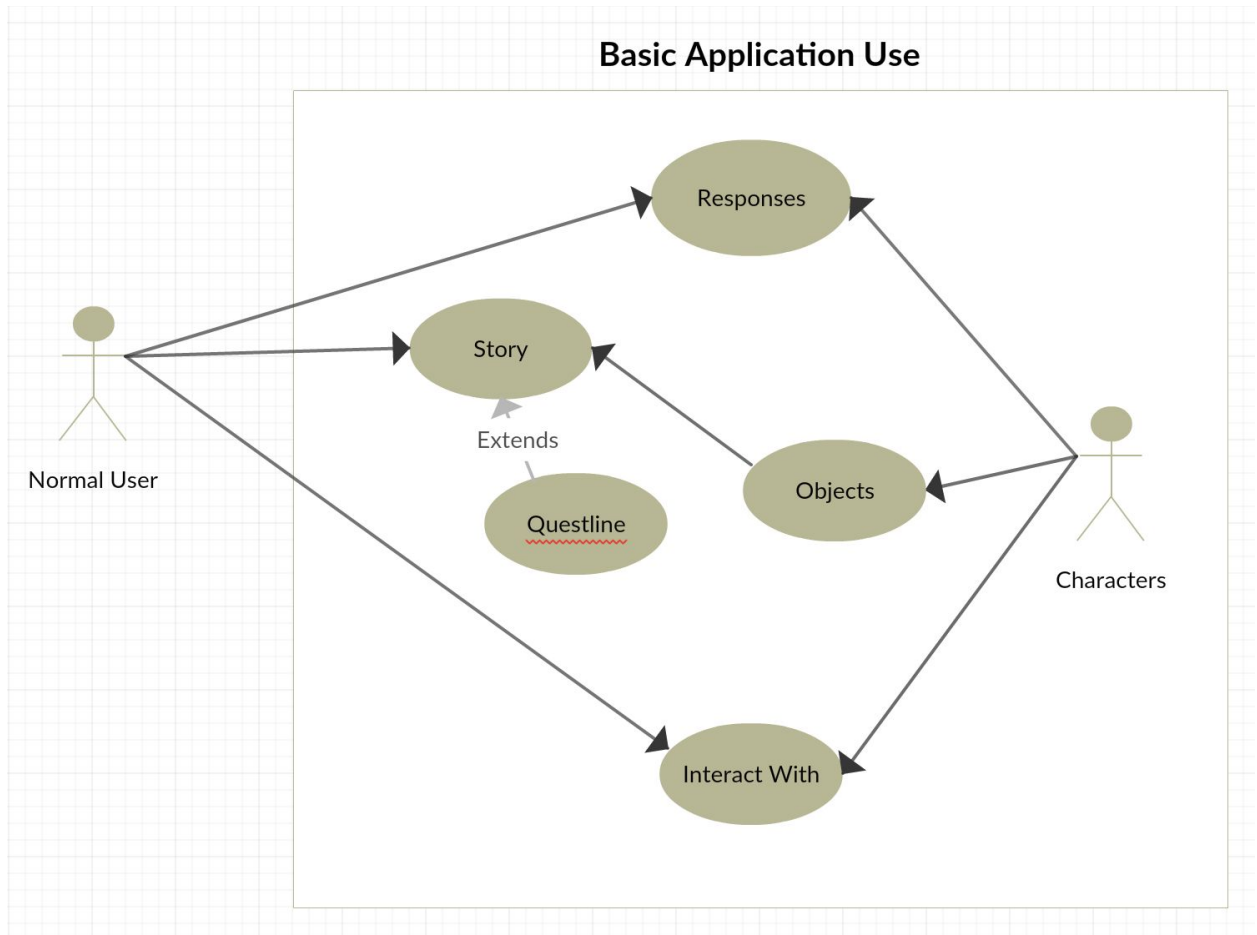
UML Diagrams

Sequence Diagram



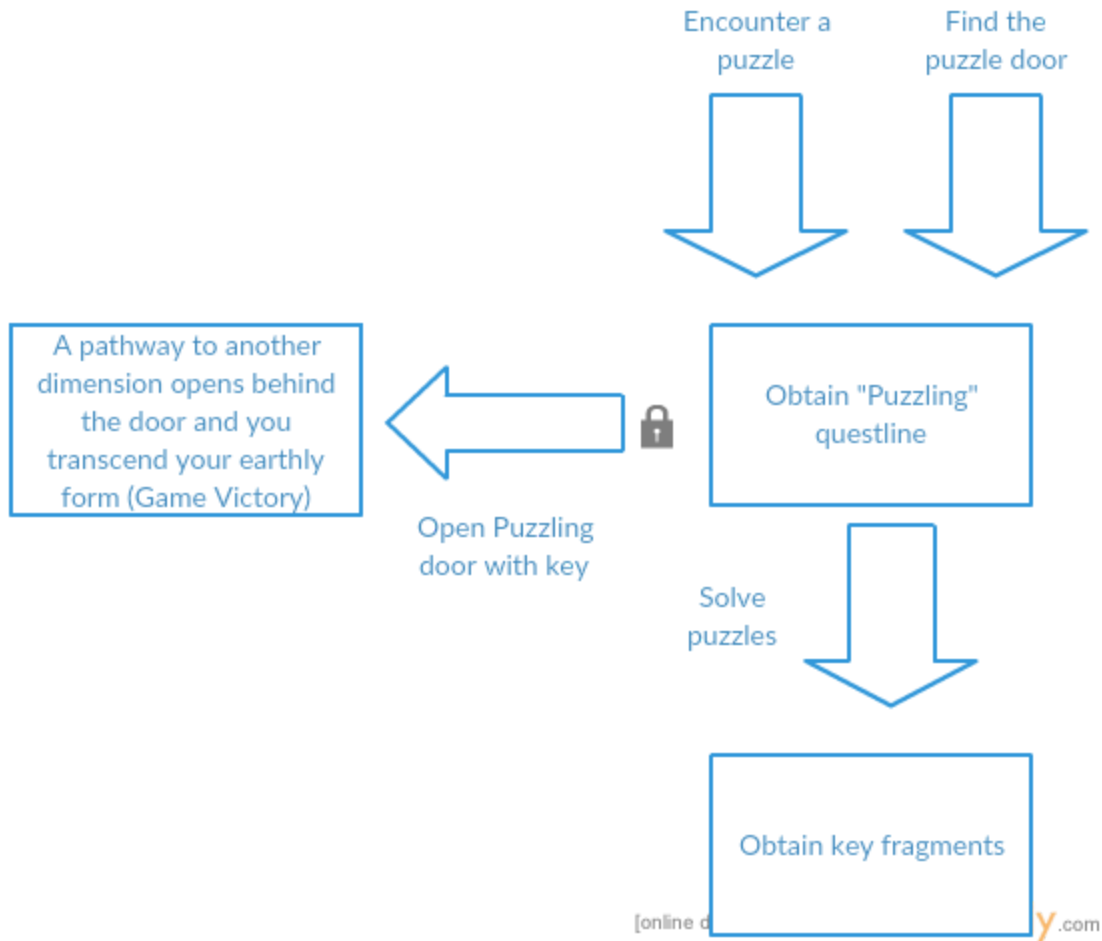
In the sequence diagram, the “Application Interface” layer provides all the options available to user for interaction with the application. The “Application Data” layer stores the data and the story line which interacts with the final hardware layer specifically the GPS. The application automatically archives data points for new locations uncovered which are pushed to the user and the “application interface” under the appropriate instances.

Use Case Diagrams



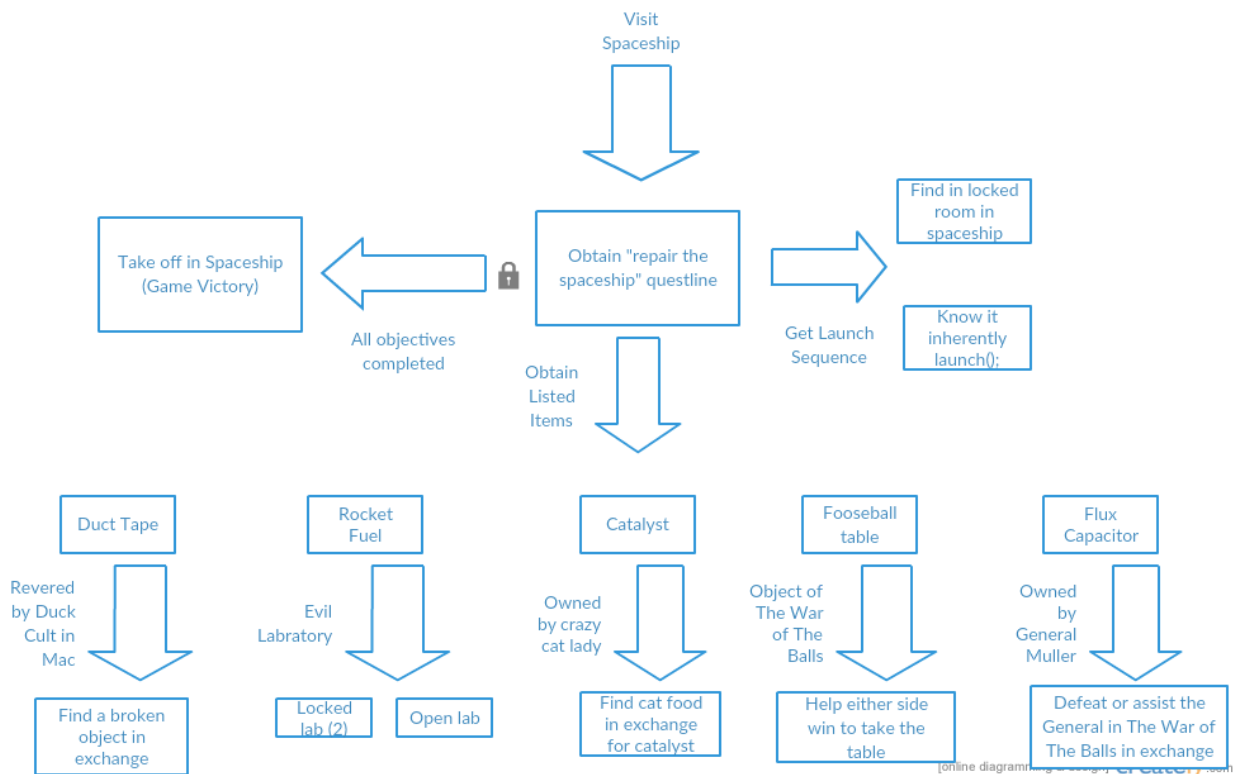
The user and the in-game characters interact in multiple ways within the application, both directly and indirectly. Directly, the user may choose options to interact with the in-game characters, like NPCs and items, via text prompts. These characters will respond to the user interaction, and the user will be prompted to interact with the object again, or be notified that the task has been completed. Indirectly, the user will interact with in-game characters by quests and quest objects. The quest objects will update quest progression.

Story Board Diagram #0



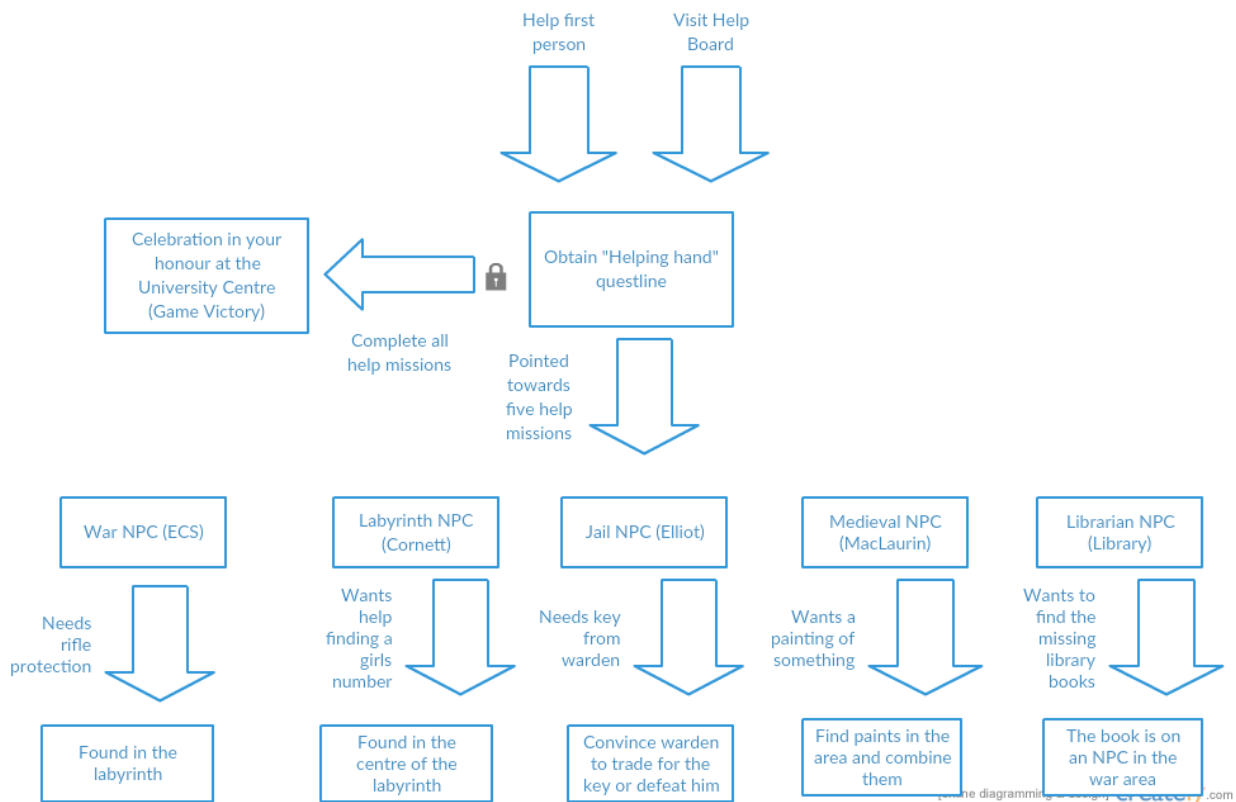
The Puzzling storyline starts with the discovery of the first puzzle or encountering the “puzzle door”. This storyline is completed when all of the fragments of the key are found, and the “puzzle door” is unlocked. The fragments of the key to the “puzzle door” are found at the completion of each of the in-game puzzles. The “puzzle door” leads to another dimension and the player transcends their earthly form.

Story Board Diagram #1



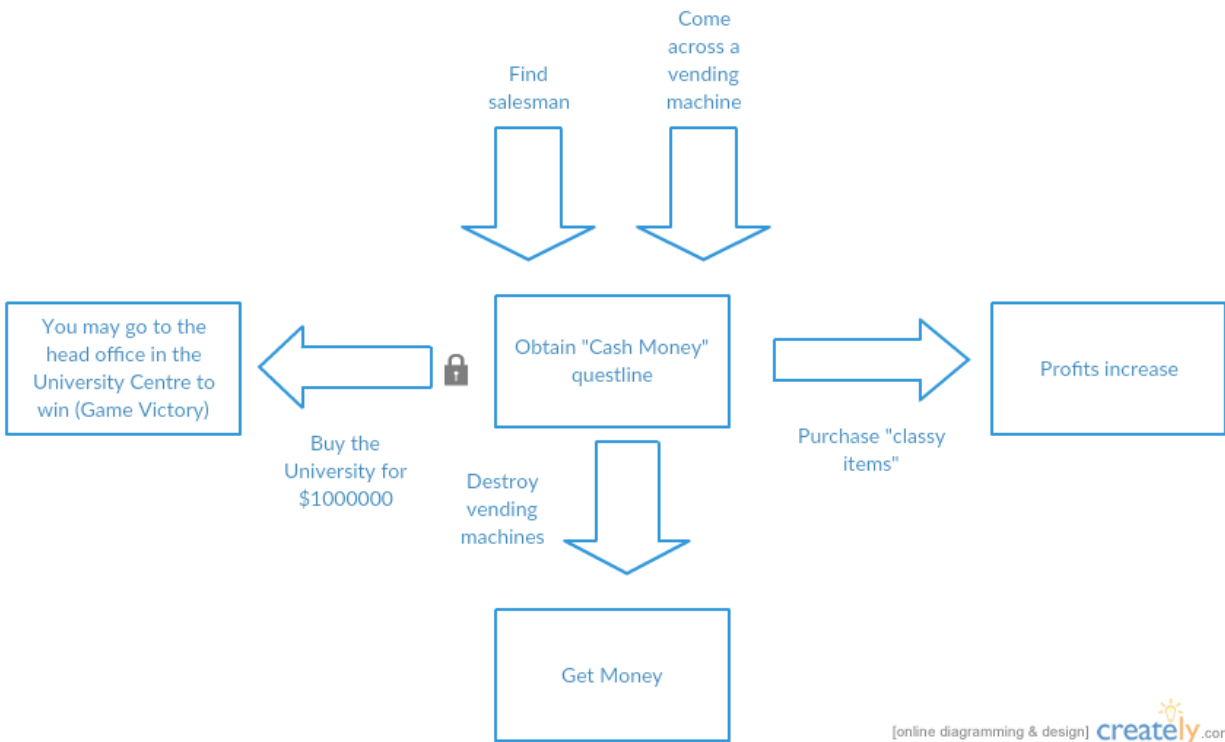
The Repair Spaceship starts with the discovery of the abandoned wreckage of a spaceship. There are two types of objects needed to repair the spaceship: launch code and equipment. The launch code has two options for discovery. The user may discover the launch code in a locked room of the spaceship or, if the user knows the launch code, they may input it into the game and launch the spaceship without discovering the launch code in the locked room. The equipment necessary to repair the spaceship may be found throughout the in-game world by exploration, talking to NPCs, solving puzzles, or completing win quests.

Story Board Diagram #2



The Puzzling storyline starts with helping the first individual or discovering of the help board. There are 5 in-game characters to help in this storyline: the War NPC, the Labyrinth NPC, the Jail NPC, the Medieval NPC, and, the Library NPC. The War NPC requires rifle protection, found at the centre of the labyrinth. The Labyrinth NPC requires a girl's phone number, found at the centre of the labyrinth. The Jail NPC requires the warden's key, obtained by convincing the warden to give the key to the player, or by defeating the warden. The Medieval NPC requires a painting, created by the player by finding and combining in-game paints. The Librarian NPC requires the missing library book, on and NPC in the war area of the game. Once all the NPCs have been helped, the player must go to the University Centre to complete the game.

Story Board Diagram #3



The Business storyline starts with the discovery of the salesman or the player discovers a vending machine. The goal of this storyline is to collect a specific amount of money by destroying the in-game vending machines. Once the specific monetary goal is complete, the user must go to the University Centre. Additionally, the player may purchase items to increase the amount of money received from the vending machines.

NPC and Item Interactions

The interaction system for the UVic escape game will be based off of the object hierarchy outlined in the UML class diagram. There are two major forms of interactions in the game world, interactions between the user and other characters, and interactions between items and characters. There is also a limited set of interactions between items and the environment.

Each character within the game will have a list of potential interactions that will be checked against to determine what has or hasn't been said and then distributed. For example, the James character can either say "Hello" or "I need some help with something. Could you help me out?" The interactions system will check priority on the interactions and find that "Hello" has a higher ranking than the asking for help string. Once "Hello" is said, the system will label it as used so the next highest priority will be shown on screen when James is next interacted with. Some strings will have a one-time use before they are eliminated from the priority system completely. The "I need help" string is an example of this.

Each item in the game also has a given set of interactions. Every item may be used in a given situation but will often have no effect. This will not destroy the item; only notify the user that there will be no effect. Otherwise each item has a table of interactions that respond to a certain set of conditions. For example, the moldy pizza may be given to an animal NPC to obtain an item. When the pizza is given it will give an indication of how the NPC interacted with the object then remove the item from the users inventory. If the user tried to give the moldy pizza to a different NPC they would likely respond by saying "no thanks", or "gross". If the user tries to give someone the pizza to complete a quest that requires a non-pizza object, they will receive the no effect notification. The item environment interactions will be set within a given items information. The item will have a special flag in the table for these interactions.

The actual data is stored and accessed using an android file saving API. It will be easily accessed through use of their data access library.

Interaction Testing



The interaction system tests will utilize a set of testing items and characters to ensure the system will correctly cycle through interactions. The testing character will be located outside the game world in an unspecified location so any users playing the game will never encounter them. Around the test character will be a number of items that can interact with the testing character, the environment, and each other. This will properly show that the cycling and listing of interaction attributes has no bugs.

Code Outline

The basic call for an interaction will be structured differently between item and character calls. The item call will look something like `getItemInteraction(String itemName, GeoLocation geolocation, String characterName)` where `itemName` and `characterName` are specified as the currently interacted with character and item. This function call will return the full text of the items interaction with the character.

The character interaction will look like this, `getCharacterInteraction(String characterName)`. The only necessary input is character name since none of our in game characters will switch between locations.

Table Structure

Character Table

Character Name	Location	GeoLocation	Interaction number
Tony Danza	Bunker 4 th floor	213 312 546	1

Character Interaction Table

Interaction Table	Flags	Description
Tony Danza	1 Norepeat	Hold me closer
	2	Hello
	3	What's going on?

Item Table

Item Name	Owned	Location	Description
Moldy Pizza	True	N/A	Flavour text
Shiv	False	3 rd Hallway Library	Flavour text

Item Interaction Table

Item Name	Character	Removal	Interaction
Moldy Pizza	Tony Danza	False	Ew this is gross
Moldy Pizza	Johnny	True	Oh thanks

Ethics

There aren't many ethical concerns when it comes to the in game interaction. The only thing to watch out for is ensuring there is no defamation of real people by characters that may or may not resemble those people. An easy way to deal with this is a notice that any similarities in name or personality a character might show has no relation to any person outside of the game world.

Data Saving & Storage

UVic Breakout, much like many RPGs, tracks in-game progression of the user. This progression is recorded in terms of storylines, side quests, items, puzzles, and other NPC and environmental interactions. There are two considerations with the progression data of the UVic Breakout: the saving of the progression data and the storage of the progression data.

Saving

There are two major types of saving styles in any video game: automatic saving and manual saving.

Automatic Saving

The first major style is automatic saving (autosaving). Using this style, the game's data is saved only at specific points in the game, or when in-game criteria is met. When the game is being saved, an icon is displayed, generally in the corner of the screen, to indicate to the user that the game is being saved. As an example, the game *Transistor*, an action RPG by Supergiant Games, uses autosaving exclusively; it cannot be saved by the user. The game saves primarily when the player transitions between areas of the game.

Manual Saving

The second major style is manual saving. Using this style, the game's data is saved exclusively by the user. That is, the the game's data is used, but not recorded, until the user indicates they want the data saved. Generally, there are multiple save "slots", locations to save data, in manual saving games. This gives a lot of choice to the user as to which data should be recorded for the next launch. This style is used primarily in games where choices are integral to the game and there are many paths for the user to take. Open world RPG games, like the *Fallout* and *Elder Scroll* series, use manual saving.

The most common style of saving for video games is a combination of both automatic and manual saving. This allows the user to control the saved data, but prevents the loss of large amounts of unsaved data.

UVic Breakout will use the autosaving style. The autosave points will be frequent to reduced any possibility to of lost data. Since it is a text-based game, implementing a manual save feature is not necessary.

Testing

To test the saving of UVic Breakout, after the save file has been created, the file will be tested for retrieval, readability, and encryption. In terms of retrieval, the test will assure that a file of the correct name and type is at the desired save location. For readability, the test will assure the save file contains the correct saved data, comparing the input saved data file and the retrieved save data file. Finally, for encryption, the test will assure the save data is readable from UVic Breakout data encryption algorithm.

Storage

UVic Breakout stores data using the File APIs from the Android developers. This API is designed for the storage of non-trivially sized collections of data. This is best for UVic Breakout since the only data to be stored is progression data, data related to the player's advancement in the game, such as story, quest, and item data. UVic Breakout will not store user specific data (like profiles or personal settings), and therefore does not need large data storage structures.

A File object is ideal for reading or writing entire collections of data, as opposed to looking for specific values within a collection of data. This is beneficial to UVic Breakout since the all of the stored data will directly affect the interactions of the player and the in-game environment.

Location

There are two main file storage locations within any Android device: “internal” and “external” storage. While removable or physically external storage is uncommon among modern devices, the physically internal storage location is partitioned into “internal” and “external” storage. With reference to UVic Breakout, this partition will be ignored and “external” storage will be considered as a storage location that is physically outside of the device, such as a removable (ie, SD card) or remote (ie, server) mediums.

External

External data is primarily used for large sets of data that is not able to be stored locally on the device or sets of data that are shared among users, multiplayer as an example. Generally, in application development, the former is not an issue. It is most certainly less of an issue for UVic Breakout as the application will store a minimal amount of data. However, the latter provides many possibilities for UVic Breakout, in terms of immediate features and scalability, but comes with many issues, primarily in security. The two major issues of using external storage are the availability and the readability of the data. An availability issue may be an issue connecting to the external storage (ie, accessing a non-existent SD card). A readability issue may be an issue with user modified data sets. These issues do not fit with the desire for UVic Breakout to be a secure and consistent application. External data storage will not be used immediately for UVic Breakout, but may be considered for future development.

Internal

Internal data storage is ideal for UVic Breakout since the application will store minimal amounts of data and does not need to share data between multiple users. Unlike external storage data, internal storage does not have the same tiers of issues with respect to availability or readability. This protects the data further from third party access, such as users or other applications. Additionally, if the user wishes to remove the application’s data, the process is much simpler and more complete with internal data storage.

For internal storage, File objects create, retrieve, write, and delete internally stored files very easily using the File() constructor, getFilesDir(), and FileOutputStream object, and delete() respectively.

Ethics

The primary ethics concern for storing data is encryption, protecting the user's data. As stated previously, UVic Breakout will not store user information or preferences, so encryption is not necessary, however, it is still valuable. UVic Breakout uses standard encryption algorithms to store data. The encryption will prevent third party alterations to game data. Users and other applications will not be able to access or alter data stored by UVic Breakout. This will provide the users with the experience without unwanted complications and to the standard outlined by the developers.



GPS

GPS is a critical technical aspect of the game. In the game, players have to move all around the UVic campus to continue with the game. This section highlights the GPS features, the implementation and testing.

GPS Features

The game revolves around the UVic campus and major buildings acting as focus of different storylines. The Android Location API for use in android app development has some very interesting features and operations it allows, however, in this document only the ones being used will be mentioned.

Determine Current Geolocation

Most Android devices allow to determine the current geolocation. This can be done via a GPS (Global Positioning System) module, via cell tower triangulation or via Wi-Fi networks.

Android contains the `android.location` package which provides the API to determine the current geo position.

Location Manager

The `LocationManager` class provides access to the Android location service. This services allows to access location providers, to register location update listeners and proximity alerts and more.

Location Provider

The LocationProvider class is the superclass of the different location providers which deliver the information about the current location. This information is stored in the Location class.

The Android device might have several LocationProvider available and you can select which one you want to use. In most cases you have the following LocationProvider available

LocationProvider	Description
network	Uses the mobile network or WI-Fi to determine the best location. Might have a higher precision in closed rooms than GPS.
gps	Use the GPS receiver in the Android device to determine the best location via satellites. Usually better precision than network.
passive	Allows to participate in location of updates of other components to save energy

Proximity Alert

In order to provide the location and trigger the story line, we will register an Intent which allows to define a proximity alert, this alert will be triggered if the device enters an area given by a longitude, latitude and radius (proximity alert).

Security



In order to access the GPS sensor, we need the ACCESS_FINE_LOCATION permission. Otherwise we need the ACCESS_COARSE_LOCATION permission.

Prompt the User to Enable GPS

The user has to enable GPS before the game begins, initially.

We will check, if a LocationManager is enabled via the `isProviderEnabled()` method, every time the game is launched. If its not enabled we send the user to the settings via an Intent with the `Settings.ACTION_LOCATION_SOURCE_SETTINGS` action for the `android.provider.Settings` class.

```
LocationManager service = (LocationManager) getSystemService(LOCATION_SERVICE);
boolean enabled = service
    .isProviderEnabled(LocationManager.GPS_PROVIDER);

// check if enabled and if not send user to the GSP settings
// Better solution would be to display a dialog and suggesting to
// go to the settings
if (!enabled) {
    Intent intent = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
    startActivity(intent);
}
```

Typically we would open an AlertDialog prompt the user and if he wants to enable GPS or if the application should be canceled.

It is not possible to enable the GPS directly in the code, the user has to do this.

Ethics

Since the game is very much integrated with location and whereabouts of the player, it is critical to follow strict guidelines for the handle of data receiving and/or storing, both locally on the device and remotely, on the servers (if any).

The Ability team compiled the following rules against the handling of user data:

- Never store data in the cloud or remotely without the permission of the users
- Present clear instructions and information to users before asking for permission to store and collect data
- Never sell data to third party without the users permission
- Allow user to opt out of the data collection

Although these set of rules can be applied to other aspects of the application and its features, there are specific rules to be followed for the handling of the location based data. There maybe be other rules that can be applied to location data.

Testing

For the testing and presentation purposes we will emulate the GPS API in the Android Studio. The android ide allows for the testing to be done within the ide and is simple enough to test different features of the application. The ide also allows the changes made during the testing and development phase to be easily transferred into the actual application when the build is formed.

Furthermore, additional tools will be used to ensure that nothing is missed.

Android Testing Support Library

This library provides a set of APIs that allow you to quickly build and run test code for your apps, including JUnit 4 and functional user interface (UI) tests. The Android Testing Support Library includes the following test automation tools:

- **AndroidJUnitRunner**: JUnit 4-compatible test runner for Android
- **Espresso**: UI testing framework; suitable for functional UI testing within an app
- **UI Automator**: UI testing framework; suitable for cross-app functional UI testing across system and installed apps

Monkey Tool

This tool runs on your emulator or device and generates pseudo-random streams of user events such as clicks, touches, or gestures, as well as a number of system-level events. You can use the Monkey tool to stress-test applications that you are developing, in a random yet repeatable manner.

monkeyrunner

This testing system provides an API for writing programs that control an Android device or emulator from outside of Android code.

Document Summary

All of the design constraints and information management for the creation of the UVic Breakout game have been listed above. The primary focus of the game is the story and puzzle solving elements, this is in line with the customers requests in the Request For Proposal of the UVic GPS based RPG. The game will use the familiar elements of UVic to build and establish a world that is heartfelt and hilarious. The secondary focus of the product is its usability and familiarity. Any University of Victoria student should be able to pick the game up and have fun with it. Each player should also be comfortable with a particular ending.

The user interface will maintain a simple and user-friendly design, building from other text-based RPG user interfaces like Zork. Images and a map will be used to show the user where they are in UVic. Many user interface elements will be reused throughout the system for consistency, such as buttons and pop ups. There will be five different screen types that can be used to implement the entire system.

NPC and item interaction for UVic Breakout is stored in a tabular format that lets our program quickly determine what interaction will be selected next. Each Item and character description is also within the tables for ease of access.

Data storage in UVic Breakout will use an android saving API that can save relatively trivial sized data extremely cheaply. This is ideal for our application because it is comprised of mostly plaintext checks for user to character interaction.

The GPS system uses the Android Location API to collect information on the user's current location. This data will be checked using a proximity alarm to switch a user's in game location. A user must have location services enabled to use our application.