ATAR Computer Science

The Smith’s Stash

Project 1 Part 2 – Development and Evaluation

Alistair Parkinson
5-17-2019

Contents

Test Plan

Alpha Testing

Alpha testing is the initial debugging that occurs during the development of the software. I apply alpha testing in three main different ways.

Playtesting

I have used playtesting in order to ensure that my code behaves as expected and that there are no logic errors in my code.

One important factor is needing to ensure when playtesting that my code works regardless of the input provided by the user. The way I test for this is by running my software, when I provide user input to the system I ensure that not only do I provide the input as expected from an average user, but I try to find inputs that break the code.

For example, as shown in Figure 1 on the left, when asked to type A, B, C or D, I type "e" as to see how the software handles the input error. The software acts as expected by re-requesting an input from the user.

Unit Testing

Unit testing is the process of isolating and testing an individual module of the system. One common way of doing this is by writing a test file, often these will call functions in the code, and test if the functions output matches with what the programmer expects. Although I didn't write a test file, I still ran functions such as, `bool HasItem(GameState gs, string item)`, in isolation, and was able to ensure that these functions worked properly returned the right values.

Additionally, I was able to unit test the processes of my code. By modifying the NewGame() module I was able to start at different points in the game. Meaning that I didn't have to playtest the entire game in order to test a single function.

Writing my Own Error Messages

One of the most common errors in my code occurred because of a variable named "gs.current". It contains a string that correlates to the function that needs to be called by main. If the "gs.current" is set to a string that does not directly correlate to a function, it will cause an infinite loop which will crash the software.

In order to combat this, I wrote an error message that would pause the software and print out the current value of "gs.current".

For example, as shown in Figure 1 on the left, I was able to recognise that the crashed because "gs.current" was set to "Lose" instead of "lose". Thanks to this error message I was able to tell what went wrong in my code before even having to look at it.

```
The blacksmith sits behind a counter. He is still rubbing the spoon with the oiled rag. He stops for a second and asks if you are looking for anything.
[Press any key to continue]
f
You currently have 2gp.

What do you do next?
A - Buy a sword for 10gp
B - Buy a bow for 7gp
C - Buy an arrow for 1gp
D - Buy a lockpicking kit for 2gp
E - Buy a pair of binoculars for 2gp
F - Buy a bottle of the smith's 'rum' for 8gp
G - Exit the armoury
G
You walk out the armoury.

You are standing outside the inn, at one of the busy streets of the city of Chadford. You are currently holding the letter that you had received from the innkeeper. Across the street is an armoury, and further down the street is a road that leads to the settlement of Sharnwick.
[Press any key to continue]
d
Now that you've bought what you need from the armoury, your feeling more prepared to take on the world.

What do you do next?
A - Visit the armoury
B - Read your letter
C - Revisit the inn
D - Drink the rum
E
What do you do next?
A - Visit the armoury
B - Read your letter
C - Revisit the inn
D - Drink the rum
d

The rum is a dark thick opaque looking liquid. You pop the cork off, and a noxious smelling odour wafts from the opening of the bottle, you try your best not to gag. By now. from the look and the smell, you are questioning yourself if it really is rum or the blacksmith is trying to poison you.
[Press any key to continue]
s
You are having second thoughts about drinking the smith's 'rum'.

Are you really sure you want to drink the 'rum'.
A - Yes
B - No
a
Reluctantly, you take a sip of the smith's 'rum'. You start to feel dizzy and then a tingling sensation coming from all over your body. Your vision stating to blur, and before you know it you have passed out.
[Press any key to continue]
p


An unexpected error has occurred.
gs.current = Lose and is an invalid value.
The software will now quit.
[Press any key to continue]
```

Figure 1A playtest done as a part of the process of alpha testing.

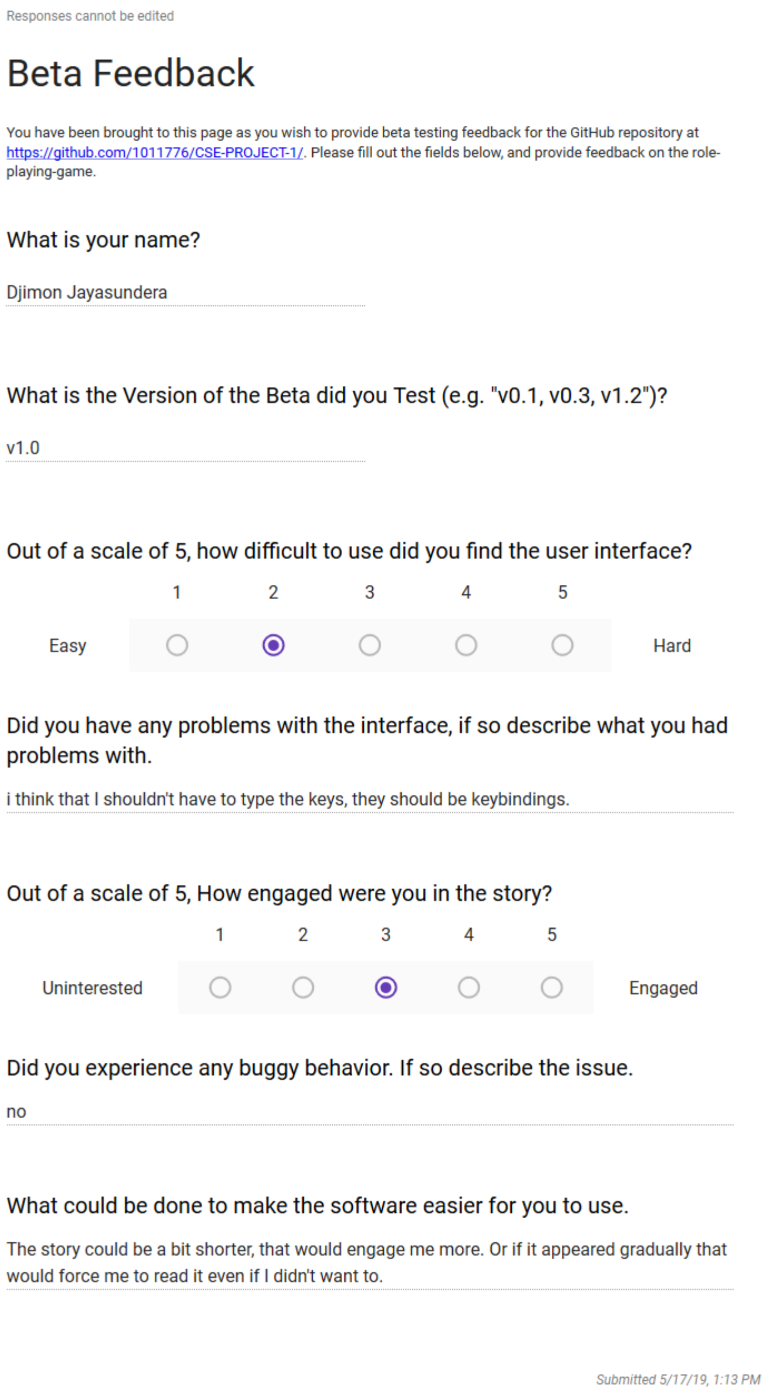
Beta Testing

I have used beta testing as a method of discovering buggy behaviour and gaining feedback, from potential users of my software, about the interface and story.

In the README.md file of my project, I have provided instructions to beta testers on how to install and run beta versions of the software. I have also provided a URL to a google form, in which the beta testers can provide feedback about the software.

The Google form provides questions to the beta testers who in turn provide feedback on the story and user interface, as well as to provide reports on any bugs in any of my beta releases.

The beta testers are required to provide the beta version number of the software tested. This is so that I can account for the changes I make, and pinpoint which issue affects which version.



Responses cannot be edited

Beta Feedback

You have been brought to this page as you wish to provide beta testing feedback for the GitHub repository at https://github.com/1011776/CSE-PROJECT-1/. Please fill out the fields below, and provide feedback on the role-playing-game.

What is your name?

Djimon Jayasundera

What is the Version of the Beta did you Test (e.g. "v0.1, v0.3, v1.2")?

v1.0

Out of a scale of 5, how difficult to use did you find the user interface?

Easy 1 [✓ 2] 3 4 5 Hard

Did you have any problems with the interface, if so describe what you had problems with.

i think that I shouldn't have to type the keys, they should be keybindings.

Out of a scale of 5, How engaged were you in the story?

Uninterested 1 2 [✓ 3] 4 5 Engaged

Did you experience any buggy behavior. If so describe the issue.

no

What could be done to make the software easier for you to use.

The story could be a bit shorter, that would engage me more. Or if it appeared gradually that would force me to read it even if I didn't want to.

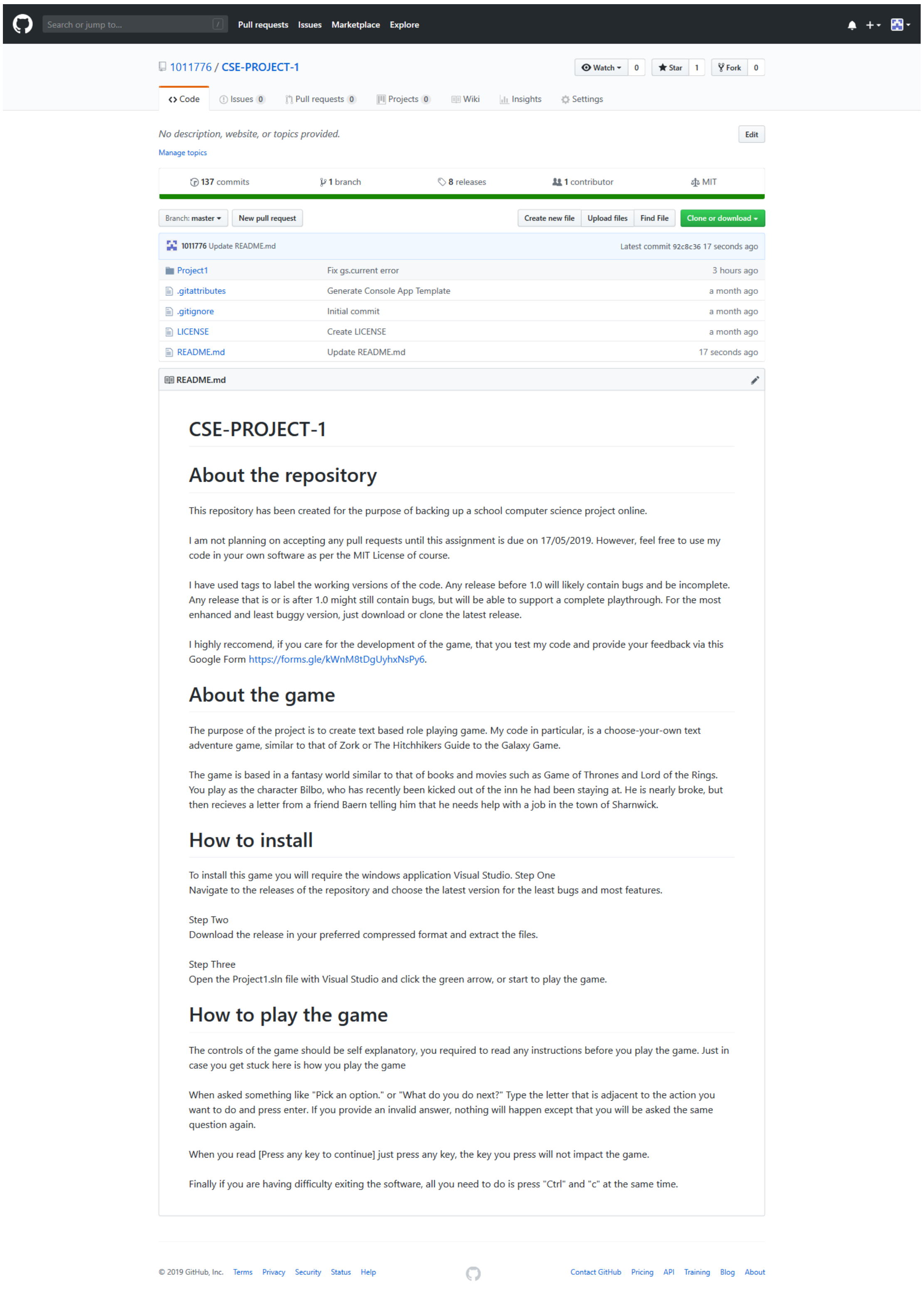
Submitted 5/17/19, 1:13 PM

Figure 2 A completed response form I received from a beta tester

On the right is one of the responses I had received about my software. From this I have taken note about what my software does well such as having a easy to use interface and lacking in bugs. But more importantly what I need to change. For example, in future updates to my software I should consider giving a typewriter effect in order to make sure that the player isn't skim reading over the story; as this could have a negative effect on the user engagement (as evident in the response form) as well as potentially causing players to miss out on key information, such as the names of people and places.

User Documentation

The documentation, as well as all the code for my project is kept on a public Github repository, at https://github.com/1011776/CSE-PROJECT-1/. All of the user documentation is contained within the README.md file. In the documentation I have outlined what the game is about, why I created the repository, how to install the game and how to play the game.


Search or jump to... Pull requests Issues Marketplace Explore

1011776 / CSE-PROJECT-1 — Watch 0 · Star 1 · Fork 0

Code · Issues 0 · Pull requests 0 · Projects 0 · Wiki · Insights · Settings

No description, website, or topics provided. Edit

Manage topics

137 commits · 1 branch · 8 releases · 1 contributor · MIT

Branch: master · New pull request · Create new file · Upload files · Find File · Clone or download

1011776 Update README.md — Latest commit 92c8c36 17 seconds ago

Project1	Fix gs.current error	3 hours ago
.gitattributes	Generate Console App Template	a month ago
.gitignore	Initial commit	a month ago
LICENSE	Create LICENSE	a month ago
README.md	Update README.md	17 seconds ago

README.md

CSE-PROJECT-1

About the repository

This repository has been created for the purpose of backing up a school computer science project online.

I am not planning on accepting any pull requests until this assignment is due on 17/05/2019. However, feel free to use my code in your own software as per the MIT License of course.

I have used tags to label the working versions of the code. Any release before 1.0 will likely contain bugs and be incomplete. Any release that is or is after 1.0 might still contain bugs, but will be able to support a complete playthrough. For the most enhanced and least buggy version, just download or clone the latest release.

I highly reccomend, if you care for the development of the game, that you test my code and provide your feedback via this Google Form https://forms.gle/kWnM8tDgUyhxNsPy6.

About the game

The purpose of the project is to create text based role playing game. My code in particular, is a choose-your-own text adventure game, similar to that of Zork or The Hitchhikers Guide to the Galaxy Game.

The game is based in a fantasy world similar to that of books and movies such as Game of Thrones and Lord of the Rings. You play as the character Bilbo, who has recently been kicked out of the inn he had been staying at. He is nearly broke, but then recieves a letter from a friend Baern telling him that he needs help with a job in the town of Sharnwick.

How to install

To install this game you will require the windows application Visual Studio. Step One
Navigate to the releases of the repository and choose the latest version for the least bugs and most features.

Step Two
Download the release in your preferred compressed format and extract the files.

Step Three
Open the Project1.sln file with Visual Studio and click the green arrow, or start to play the game.

How to play the game

The controls of the game should be self explanatory, you required to read any instructions before you play the game. Just in case you get stuck here is how you play the game

When asked something like "Pick an option." or "What do you do next?" Type the letter that is adjacent to the action you want to do and press enter. If you provide an invalid answer, nothing will happen except that you will be asked the same question again.

When you read [Press any key to continue] just press any key, the key you press will not impact the game.

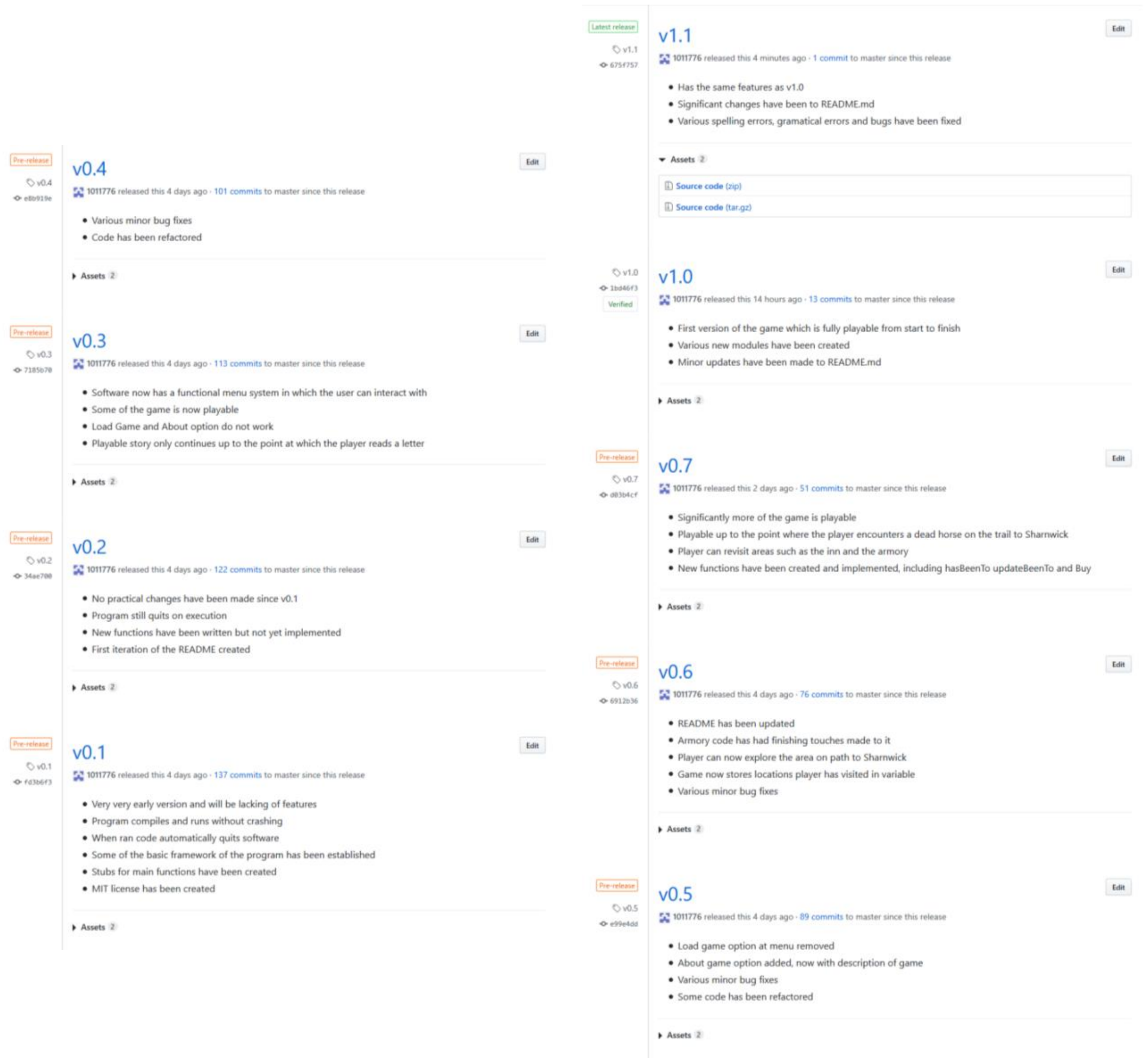
Finally if you are having difficulty exiting the software, all you need to do is press "Ctrl" and "c" at the same time.

© 2019 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub Pricing API Training Blog About

Figure 3 The GitHub repository containing the code and documentation

Every time I make significant progress to the project I create a new release and describe what was features were changed, as to keep users up to date with the latest working version of the code.

I have used v0 to label releases in which players are not able to complete the game from start to finish. v1 releases on the other hand mean that the game is playable from start to finish.



Latest release
v1.1
675f757

v1.1

1011776 released this 4 minutes ago · 1 commit to master since this release

- Has the same features as v1.0
- Significant changes have been to README.md
- Various spelling errors, gramatical errors and bugs have been fixed

Assets 2

Source code (zip)

Source code (tar.gz)

v1.0
1bd46f3
Verified

v1.0

1011776 released this 14 hours ago · 13 commits to master since this release

- First version of the game which is fully playable from start to finish
- Various new modules have been created
- Minor updates have been made to README.md

Assets 2

Pre-release
v0.7
d83b4cf

v0.7

1011776 released this 2 days ago · 51 commits to master since this release

- Significantly more of the game is playable
- Playable up to the point where the player encounters a dead horse on the trail to Sharnwick
- Player can revisit areas such as the inn and the armory
- New functions have been created and implemented, including hasBeenTo updateBeenTo and Buy

Assets 2

Pre-release
v0.6
6912b36

v0.6

1011776 released this 4 days ago · 76 commits to master since this release

- README has been updated
- Armory code has had finishing touches made to it
- Player can now explore the area on path to Sharnwick
- Game now stores locations player has visited in variable
- Various minor bug fixes

Assets 2

Pre-release
v0.5
e99e4dd

v0.5

1011776 released this 4 days ago · 89 commits to master since this release

- Load game option at menu removed
- About game option added, now with description of game
- Various minor bug fixes
- Some code has been refactored

Assets 2

Pre-release
v0.4
e8b919e

v0.4

1011776 released this 4 days ago · 101 commits to master since this release

- Various minor bug fixes
- Code has been refactored

Assets 2

Pre-release
v0.3
7185b70

v0.3

1011776 released this 4 days ago · 113 commits to master since this release

- Software now has a functional menu system in which the user can interact with
- Some of the game is now playable
- Load Game and About option do not work
- Playable story only continues up to the point at which the player reads a letter

Assets 2

Pre-release
v0.2
34ae700

v0.2

1011776 released this 4 days ago · 122 commits to master since this release

- No practical changes have been made since v0.1
- Program still quits on execution
- New functions have been written but not yet implemented
- First iteration of the README created

Assets 2

Pre-release
v0.1
fd3b6f3

v0.1

1011776 released this 4 days ago · 137 commits to master since this release

- Very very early version and will be lacking of features
- Program compiles and runs without crashing
- When ran code automatically quits software
- Some of the basic framework of the program has been established
- Stubs for main functions have been created
- MIT license has been created

Assets 2

Figure 4 List of all of the releases made on GitHub

Evaluation

Design Evaluation

Overall, I have made an effort to stick to my original design, as to limit my time waste time on feature creep.

Of the 11 functions that I came up with during the design stage of the project. I have implemented all of them in my proper C# code. I have also created 12 other functions, most of them, however are essentially the same as the original functions but more broken down. Below is a table of all the functions I included in my code:

Comparison of modules from design to modules in final product

Function in C# Code	Was in pseudocode	Brief Description
`void Main(string[] args)`	Yes	Presents player with a title screen and options to start the game and read the about menu.

Processes called by the switch statement in main (Functions for areas and events)

Function in C# Code	Was in pseudocode	Brief Description
`void Menu(GameState gs)`	No	Presents player with a title screen and options to start the game and read the about menu.
`void About(GameState gs)`	No	Prints to user information about the game.
`void Inn(GameState gs)`	Yes	Main character is kicked out of an inn and is given a letter from Baern
`void Letter(GameState gs)`	Yes	Main character reads the letter given to him by the innkeeper.
`void Chadford(GameState gs)`	No	Prints description of the streets of Chadford. Gives player options about what to do next.
`void Armoury(GameState gs)`	Yes	Player can buy items from armoury or return to the streets of Chadford.
`void DeadHorse(GameState gs)`	Yes	The main character stumbles by dead horses sprawled across the road.
`void Sneak(GameState gs)`	No	The main character attempts to sneak up on goblin ambushers.
`void Attack(GameState gs)`	No	The main character attempts to attack goblin ambushers.
`void Surrounds(GameState gs)`	No	The main character checks surroundings at the site of the dead horses.
`void Inspect(GameState gs)`	No	The main character has a closer look at the dead horse.
`void Ambushed(GameState gs)`	No	The main character is attacked by surprise by goblin ambushers.
`void Sharnwick(GameState gs)`	Yes	The main character arrives at Sharnwick but is forced to turn back to look for Baern.
`void Rum(GameState gs)`	No	The main character takes a sip of the smith's rum and is poisoned.
`void Hideout(GameState gs)`	Yes	The main character has followed the goblin ambushers to their hideout.

`void Captured(GameState gs)`	Yes	The main character has been captured by the goblin ambushers
`void Escape(GameState gs)`	No	The main character uses the lockpicking kit to escape the cell, but gets thrown back in due to being unarmed.
`void RumGuard(GameState gs)`	No	The main character gives the rum to the goblin guard and is able to escape.
`void Win(GameState gs)`	Yes	When the character rescues Baern, the player is taken to the win screen.
`void Lose(GameState gs)`	Yes	Whenever the main character is captured or dies, the player is taken to the lose screen.

Modules not called by the switch statement in main

Function in C# Code	Was in pseudocode	Brief Description
`GameState NewGame()`	Yes	Returns GameState for a new game.
`bool HasItem(GameState gs, string item)`	Yes	Returns whether the gs.inventory contains item using linear search algorithm.
`void Pause()`	No	Prompts player to press any key to continue.
`bool HasBeenTo(GameState gs, string section)`	No	Returns whether the gs.hasBeenTo contains section using linear search algorithm.
`void UpdateBeenTo(GameState gs)`	No	Updates the gs.beenTo variable
`void Buy(GameState gs, string item, int cost)`	No	Prints appropriate message and manages gs.gp as well as gs.inventory when main character tries to buy an item.
`int PresentOptions(List<string> options, string message)`	No	Prints out options and prompts user to select one, message is printed with options presented.

Although I have significantly more functions than the initial design. Most of these functions are simply the same, as the ones in the design, but have broken down into lower levels of abstraction.

Take for example the ambush process. This process in my pseudocode handles the events from stumbling across the dead horse on the road, leading up to the main character either being captured or finding the goblin hideout. In my C# code I have broken this down into the sub processes:

- `void DeadHorse(GameState gs)`
- `void Sneak(GameState gs)`
- `void Attack(GameState gs)`
- `void Surrounds(GameState gs)`
- `void Inspect(GameState gs)`
- `void Sneak(GameState gs)`
- `void Ambushed(GameState gs)`

During playtesting, I noticed that if I skim read or didn't read the story, the game could be completed in under minute. At this point I decided to make a few changes to my original design.

First of which was to create the Pause function. The pause function prints "[Press any key to continue]" and requires the player to press a key. I put this in between long print statements as to prevent players from skim story, by slowing down the pace in which the player reads the story.

Another change I made was increasing the difficulty of game. Unlike my original flow chart which included more ways to win the game. I had decided to increase the difficulty of my game, so that the player must make more attempts at playing my game.

In my here are only three ways to win my game:

- Buying one bow, one arrow and a pair of binoculars. Using the binoculars spot the goblin ambushers, then instead of attacking them with a bow, you follow them to Baern's cell and shoot the goblin guard who is carrying the keys to Baern's cell.
- Buy the smiths rum, then get captured, not killed, by the goblins. Then give the rum to goblin guard, so that it drops its keys. You then use the keys to free Baern.

I intentionally chose these two win conditions as opposed to something more obvious such as buying the sword or trying to use the lockpicking kit on the cell, as to make the game more difficult.

The final change I made to my original design, was to include a menu system where the player has the option to read what the game is about. This was simply done to make the game seem more professional and familiar to people who do play text-adventures which normally have menus and title screens.

Almost all of the criteria outlined in the analysis have been met to a high standard. I have highlighted the criteria that are strongly met in green, and not met in red.

In terms of the user interface all of the criteria have been met to a high standard:

- The user interface is does not need external user documentation or previous experience in playing text adventures.
- The user interface is scalable to different monitor and window sizes (as long as user changes the console font size appropriately)
- The input is minimised as only two keyboard presses are required per action and one keyboard press to continue after a pause.
- The user interface handles input errors in a way that will not confuse or frustrate users (user interface simply prompts the user to re-type input)
- No unnecessary output is printed to the screen as it does not print out the game state every action but instead flavourful text to aid in the immersivity of the game.

In terms of user needs, the criteria has been met to a medium standard:

- The user must feel immersed in the world and character that they play, thus the player should feel a sense of control over their character and that their decisions impact the world in which the game is based. However according to beta testers this is not the case as some are uninterested in the story.

- The game keeps the user engaged and strikes a balance between being challenging and rewarding. As discussed earlier the increased difficulty has made my game take longer but
- The game should be reliable and bug free as not to break immersion. This is true as I have not yet ran into any bugs in my latest release of the project.

In terms of the criteria surrounding potential users, most criteria have been met. :

- I have distinguished and made it clear what code works and doesn't work
- This in turn implies that I have have warned users of potentially damaging code
- I have internally documenting my software well
- My installation guide has made the process installation process user friendly, but only if they have Visual Studio

In terms of legal obligations I have made sure code abides with all relevant acts:

- The Copyright Act (1968) and Copyright Amendment (Digital Agenda) Act (2000) has been met, as my code does not contain any copyrighted material.
- Privacy Act (1988) and Privacy Act (2000) has been met as I do not store any personal user information.
- The Spam Act (2003) has been met as my software does not send any electronic messages to the user.

In terms of ethical obligations, all of my criteria have been met to a high standard:

- My software is safe to use and not contain security flaws, even though I do not hold liability if something does happen.
- My software does not purposely offend any of my users.
- My software does not unfairly take advantage over its users.

Overall, the number achieved criteria greatly outnumbers the number of criteria that were not achieved. However, two criteria that were not achieved, were in some way or another, subjective. In conclusion, although two criteria were not fully met, the project has been done to an adequate standard.

Design Evaluation

After a stringent alpha and beta testing stage, I believe most bugs in the system have been identified. And of all the bugs that have been identified, all of them have been patched. This however does not mean that software is bug free; but rather this means that the system is most likely bug free.

After the process of identifying and removing bugs, I have noticed a pattern to which are the most frequent. As discussed earlier in the alpha testing section, a reoccurring problem is that if the variable gs.current is set to an invalid it will cause a customised error to appear which will print out what value of gs.current caused the crash. Often the solution to this is simply to identify all instances (often one) of the incorrect value and replace it with the correct one. As more modules are added to the project the more this bug is likely to reappear as a new string needs to be created to correspond to their new modules.

The impact of this bug is moderate. The impact is not low because it stops the user from progressing in the game. However, the impact is not high, as it does not put the user's security, privacy or safety at any risk. Thus, if this bug ever arises anywhere in the code, the appropriate action of hot fixing the bug, should be taken.

According to the feedback from the beta testers. To improve the user experience. More work needs to be done about rounding out the story and making it more engaging. By doing this it could have a crucial impact on the system's performance as it would have positively impact on the users' experience.

Although all of the features that I have outlined in the design stage have been met. There are still many features which I am yet to implement. Possible features could include:

- Switching to a GUI from a console by using the WPF application framework as opposed to using a console application. As the implementation of buttons is arguably more intuitive and familiar to most users than a console.
- Slowing down the rate at which text appears, as opposed using `void Pause()` to stop players from skim reading important text. This would decrease the input needed from the user.
- Introducing a combat system, this would make fights in the game seem fairer, as opposed to being destined to lose or win the fight before it has started. This however could also have a negatively impact the game as it would disrupt the choose-your-own aesthetic of the game.
- Making the software compatible with systems that don't have Visual Studio and with systems with different operating systems apart from windows. This would make the system more accessible.

References

Parkinson, A. (2019). Google Forms – create and analyse surveys, for free. Retrieved from https://forms.gle/kWnM8tDgUyhxNsPy6

Parkinson, A. (2019). 1011776/CSE-PROJECT-1. Retrieved from https://github.com/1011776/CSE-PROJECT-1

Programmes interactive. Retrieved from https://downloads.bbc.co.uk/interactive/embed/container.html?url=//downloads.bbc.co.uk/interactive/h2g2/main.js&height=577px&width=944px&path=//downloads.bbc.co.uk/radio/games/h2g2/

Zork. Retrieved from https://en.wikipedia.org/wiki/Zork