**Basic Reb0rn**

**Technical Design Document**

*This technical design document describes the technical details and requirements for BASIC Reb0rn, a text adventure game ported from AppleSoft BASIC with a Haunted House theme.*

# Contents

[Contents 2](#_Toc21470091)

[Section 1 – Game Overview 3](#_Toc21470092)

[1.1 Game Summary 3](#_Toc21470093)

[1.2 Platform 3](#_Toc21470094)

[Section 2 – Development Overview 4](#_Toc21470095)

[2.1 Development Environment 4](#_Toc21470096)

[2.2.1 Development Hardware 4](#_Toc21470097)

[2.2.2 Development Software 4](#_Toc21470098)

[2.2.3 External Code 4](#_Toc21470099)

[Section 3 – Game Mechanics 5](#_Toc21470100)

[3.1 Main Technical Requirements 5](#_Toc21470101)

[3.2 Architecture 5](#_Toc21470102)

[3.3 Game Flow 6](#_Toc21470103)

[3.4 Game Objects and Logic 7](#_Toc21470104)

[3.5 Data Management and Flow 7](#_Toc21470105)

[Section 4 – User Interface 8](#_Toc21470106)

[4.1 Game Shell 8](#_Toc21470107)

[4.2 Play Screen 9](#_Toc21470108)

# Section 1 – Game Overview

# 1.1 Game Summary

*Basic Reb0rn is a text adventure game based off of Haunted House in the* [***Write Your Own Adventure Programs for Your Microcomputer***](https://www.amazon.co.uk/Write-Your-Own-Adventure-Microcomputer/dp/0686878329)*book. Players must make their way through a haunted house, discover its secrets and eventually escape. Locations in the house will have obstacles, objects or secrets for players to find, use and navigate.*

## 1.2 Platform

*The game is a text adventure and so is most suited to keyboards, therefore the game will be made for Windows/Mac/Linux.*

# Section 2 – Development Overview

## 2.1 Development Environment

*The game will be made using the* [***ASGE Game Framework***](https://huxyuk.github.io/AwesomeSauceGE/getting-started.html) *and C++.*

### 2.2.1 Development Hardware

*The game will be developed on Windows computers.*

* *CLion requires a minimum of Windows 7, 2GB of RAM and 2.5GB of disk space (plus 1GB for caches).*
* *GitKraken requires a minimum of Windows 7, 4GB of RAM and 5GB of disk space.*

### 2.2.2 Development Software

*I will be using* [***CLion***](https://www.jetbrains.com/clion/) *to develop C++ code. Version control will be handled by* [***Git***](https://git-scm.com/) *with the repository hosted by* [***GitHub***](https://github.com/)*. I will be using*[***GitKraken***](https://www.gitkraken.com/) *to provide a graphical interface for Git.*

### 2.2.3 External Code

*I will be using the ASGE Framework for C++ written by James Huxtable to provide basic game functionality such as the game loop, sprite rendering and input events.*

*I will also be using the* [***Json for Modern C++***](https://github.com/nlohmann/json) *library by nlohmann for loading in data from JSON files.*

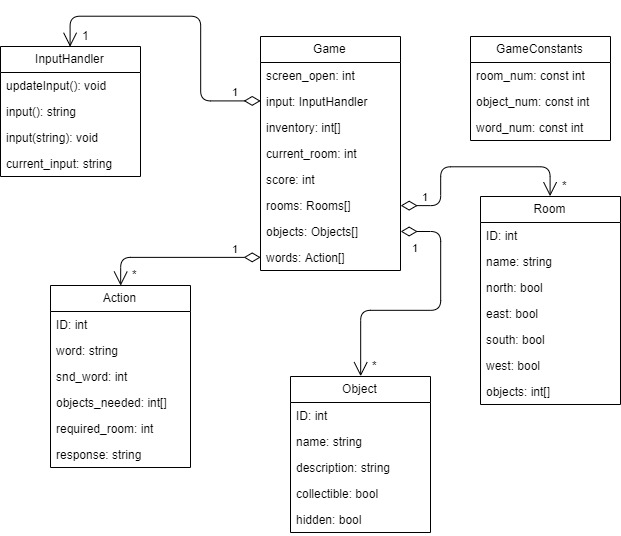
# Section 3 – Game Mechanics

## 3.1 Main Technical Requirements

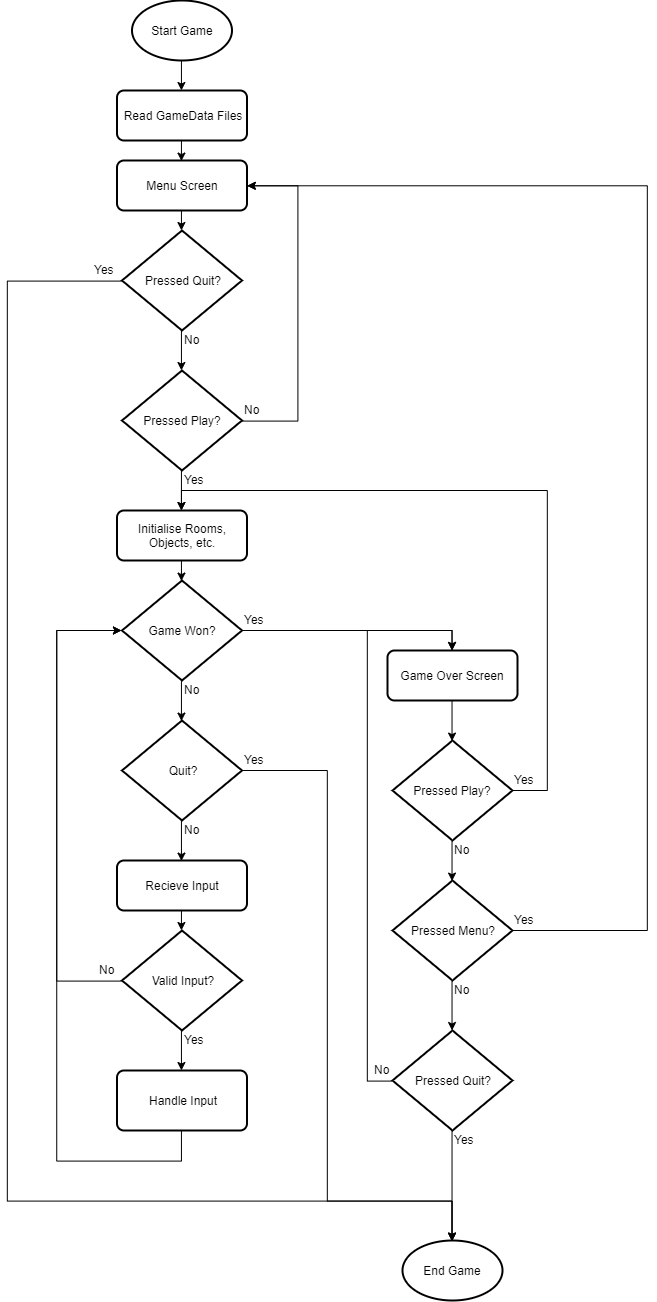
*The main technical requirement for this game will be finding a suitable way load and read game files. There are several ways this could be done: txt, json, xml etc. I have decided to use Jason since it is easy to read and will be simple to implement with the* [***Json for Modern C++***](https://github.com/nlohmann/json/) *library.*

*The rest of the game will involve setting up: movement between rooms, gathering player input and performing actions.*

## 3.2 Architecture

**

## 3.3 Game Flow



## 3.4 Game Objects and Logic

***Room***

*Rooms in the game will house information such as ID, name, directions and any items. The player will be able to leave the room depending on the directions set and collect any item that is visible in the room. They player will also be able to leave a maximum of 5 items in a room before they are told there is not enough space.*

***Object***

*Objects in the game will have an ID, name, description (for use with EXAMINE), collectible and hidden. The hidden objects will be revealed by using EXAMINE. E.g. using EXAMINE COAT in the CUPBOARD will reveal the hidden KEY.*

***Action***

*Actions will have an ID, verb, optional second word, the objects needed, the required room, and a response for when the action is completed. The results of these actions will be dealt with by the Game, this class will be just to hold information about the different actions.*

***Input Handler***

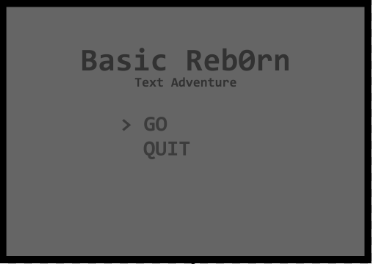
*The input handler will take information from the key presses and edit a string holding the current input.*

## 3.5 Data Management and Flow

*The game data (rooms, objects and actions) will need to be loaded in from a file to be easily editable. Both C++ and ASGE have their own methods for file handling, I will be using ASGE’s version.*

# Section 4 – User Interface

## 4.1 Game Shell



*The two non-gameplay screens of the game are the Main Menu and the Game Over screens. Users will select the option they want by using the arrow keys to move up and down between the selections and then pressing enter to confirm their choice.*

***Main Menu***

*The main menu displays the game title and 2 options:*

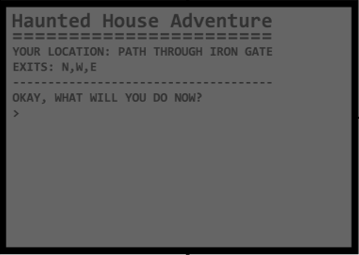
* ***GO:*** *Will allow the player to start the game on selection*
* ***Quit:*** *Exits the program*

***Game Over***

*The game over menu will display the player’s score as well as the following options:*

* ***Play Again:*** *Will reset the game, allowing the player to play again*
* ***Menu:*** *Will send the player to the main menu*
* ***Quit:*** *Exits the program*

## 4.2 Play Screen



*The main game will have a resolution of 1024px wide by 768px tall as this is a suitable size for PC monitors.*

***Main Game Screen***

*The main game screen will have the title at the top with the game information (location, exits, items) displayed below it. Below all the information will be a space for the player to input their text. Feedback will be given below the player input and the information above will be updated accordingly.*