Computer Science Project: Story Teller

By Nathan Windisch

Contents

[Analysis 2](#_Toc461134287)

[Design 4](#_Toc461134288)

[Technical Solution 7](#_Toc461134289)

[Testing 8](#_Toc461134290)

[Evaluation 9](#_Toc461134291)

[Code 10](#_Toc461134292)

[Bibliography 20](#_Toc461134293)

# Analysis

The product that I am creating is a text-based application that allows users to play a type of game called a gamebook where users read the story and can make decisions that change the outcome of the story. When completed this application will be very complex and will have many stories, each with multiple endings based on how the user plays the game. Expansion upgrades will be available in the form of micro-transactions, but these could also be unlocked either by watching adverts or completing achievements, both of which will give the user a soft currency. Cosmetic upgrades such as app skins could be implemented, for easier reading at night with a dark theme.



Figure I: A dark theme for an Android device, similar to what would try to achieve with our product.

This application was commissioned by Oliver Cox, a local entrepreneur who had the main idea for the project. He has an idea for the first story line that will be used in the application and the vocals that will accompany the story. The application was originally going to be sold on the Steam (PC Version) and Google Play/Apple App Store (Mobile Version) for £5, but will have the aforementioned micro-transactions built in. Upgrades will also be available that will allow users to get rid of advertisements.

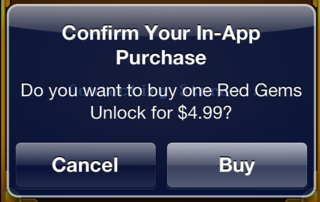


Figure II: An example of Micro-transactions in a game on the Apple store.

The user will use the game to be able to read though (and/or listen to, in the future) stories that have been prewritten. Every time a decision is required, the user will be able to make a choice that will change the outcome of the story. An example of some things that they can do is they can choose if they should release a dragon from its cage to steal its treasure; killing all of the people in a nearby town, or whether they should slay the dragon and give the gold to the townsfolk.

Who is the client? Oli Eastgreen Cox is the person who has requested the game.

What do they want? A PC application that allows users to play Gamebooks.

How much time do we have? 2 months.

How is the client going to use it? Sell it on Steam for £5 and upgrades for more money.

How is the user going to use it? The user can access stories and read though them, choosing how their character plays as they read. They can either read on their own (or have a narrator read along with them, in the future). When they make a choice a new set of text pops up, continuing the story from the choices they make. Once the user finishes the story, either by completing the main objective or by dying, the story will end. The user will then be rewarded with XP and a soft currency, the latter of which can be used to purchase more stories. The user can also look back on previous stories and share them with their friends and family.

Why does the user want the product? Users may want the product as it enables a desire to read and learn, resulting in an expanded knowledge and vocabulary. The product will also appeal to those who enjoy fantasy and adventure books as they can get lost in the story.

What does it need to be successful? Good storyline will be required in order to capture the reader’s attention. A well designed User Interface will be needed in order to generate easy to use user interaction.

The main issue with the market in which the application is going to be launched into is that there are not many story based mobile applications out there, meaning that our game will be released into unclaimed waters. This should mean that there are many people who want a product that we are selling, as there is nothing else quite like it. Our application should spawn lots of other competition as people realize that our market that we have tapped into is a good one, resulting in better products and more variety for the end users.

# Design

The design of the application was originally going to be an Android application, but given the time and budget constraints, I decided to change it into a text-based adventure game to ensure that a prototype of the product would be available on time.

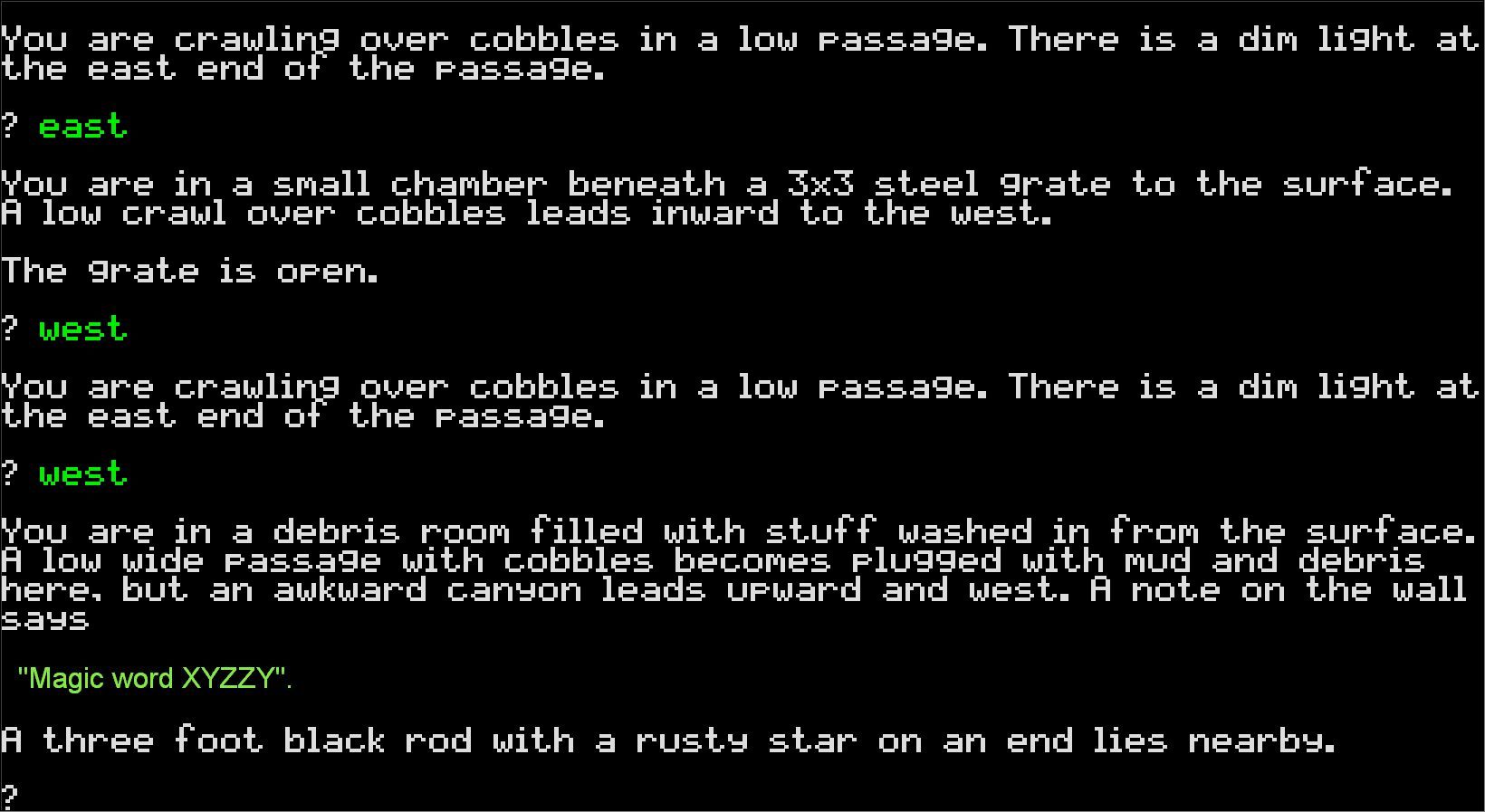


Figure III: An Example of a Text Based Adventure Game - Colossal Cave Adventure

The design of the game is similar to that of text-based games made in the 1980s, such as Alter Ego, Rogue, Beast, The Hitchhiker’s Guide to the Galaxy, MIST and Zork. The program will be written in C# and will use the basic libraries and APIs that come with the language, but more APIs will be added if/when needed. The program is being built to allow players to play video games whilst increasing the reading proficiencies, making them better and faster readers whilst giving them a fun experience.

The data that the user needs to store, such as configuration and statistics, shall be stored in a flat-file format, using XML as the file format used to generate and store that data. I have decided to do this as I feel that a flat-file will be easier to generate, manage and edit than databases such as MySQL, MS Access and SQLite, and I have chosen XML as I find it easier to read and understand, when compared to its counterparts such as JSON or CSV.

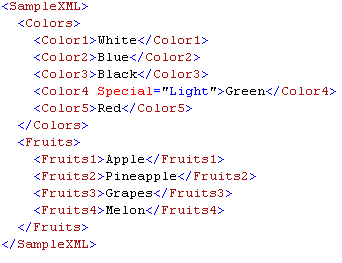


Figure IV: An Example of an XML File

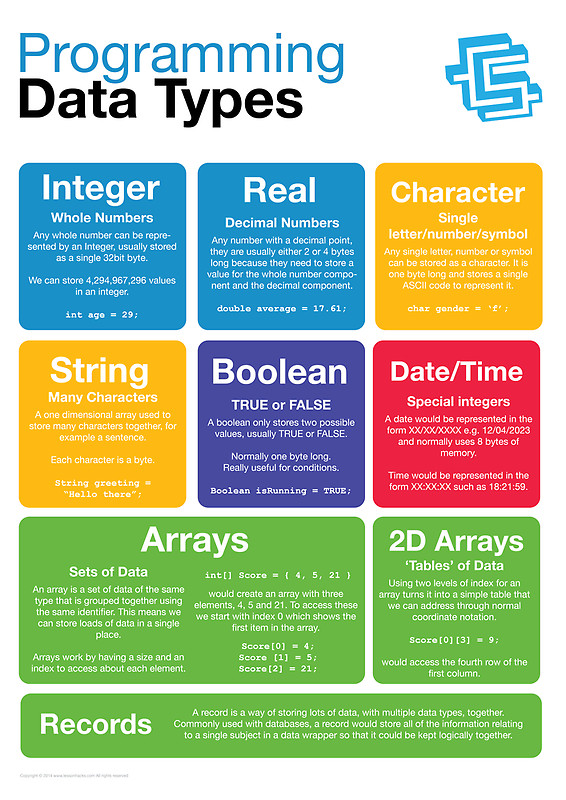
I have chosen to use a variety of primitive data types for my project, including Strings, Characters, Integers, Floats/Doubles and Booleans. I have specifically chosen each variable’s data type to ensure maximum efficiency when transferring data. 

Figure V: A list of datatypes with explanations.

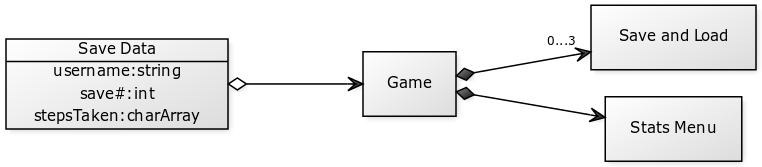
The original program was written in Java as an Android application, but given that I don’t have an extensive knowledge of Android application development, I ended development on the project. The second prototype is the current version of the program. It is written in C# and is much better than the original prototype. I have only briefly worked on the project at this moment in time, but it has a functioning menu and framework has been laid out for a settings menu as well. A brief UML diagram has been generated to show how the save data will be saved can be seen below.

Figure VI: Save Data UML Diagram

Below is a flow chart of how the game will run, from the user’s perspective.

C:\Users\Natfan\Downloads\userPerspectiveGameRun.png

Figure VII: A brief flow chat of the user's experience.

As shown above, the program will allow the user to play the game or edit settings. See below for a more detailed flow chart of how the game will run.

**<TODO: MAKE DETAILED FLOWCHART>**

# Technical Solution

The Config class is used to store all of the configuration variables that are used in the program. Most of the variables are stored within the accompanying App.config file which is using an XML format, but the Text Frequency variable is not due to debugging purposes.

The Program class is where the bulk of the project is. The variable ‘noType’ is used to denote whether the text should be typed out character by character or whether it should just be printed. The variable ‘ms’ is used as a mathematical figure to calculate how fast the typing is, and should not be changed. Many variables within the Program class are just set to their configuration counterparts, or are used as storing commonly used text to save time and space.

Within the ‘Main’ segment of the Program class, the Console is cleared and the program is run, where the Title of the window is set, along with the size and colour. After

# Testing

When completing the testing part of my project I had to complete several in-depth tests of my product to ensure that there were as little bugs in the prototype as possible. Some of the tests that I performed were:

* Complete the game with all scenarios.
* Complete the game with settings changes such as fastType and instaType enabled in different tests.
* Edit settings in-game.
* Give the game to other people in order to see if they do anything differently.

When performing these tests, I found a few bugs. These are as follows, in order of the tests aforementioned.

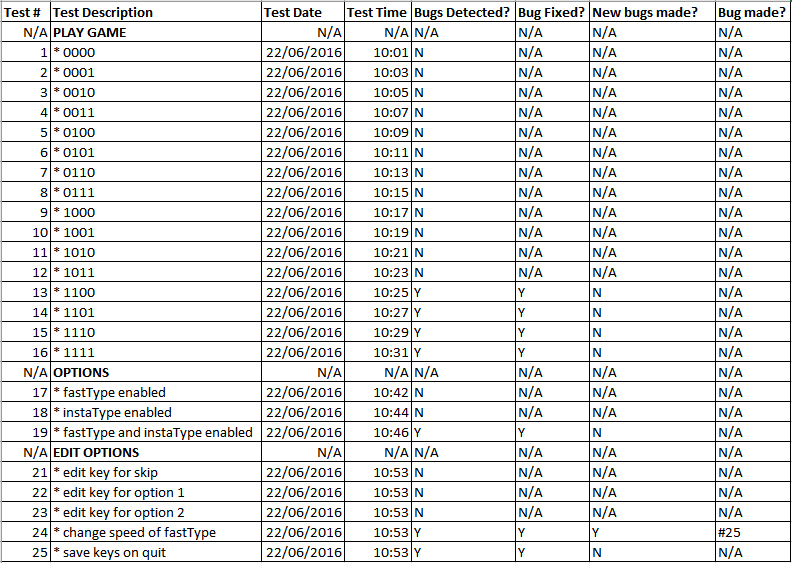
* The game could not be completed when going through the second pathway. This was because the key that was required to open the door was not placed in the room.
* When playing the game with both fastType and instaType on, the text would flicker rapidly.
* The keyboard binds could not be edited in game.
* The speed of typing (fastType) could not be edited.

After these bugs were found, I attempted to roll out fixes for them. The following is a list of fixes that I made, in order of the aforementioned bugs listed.

* Added a key to the second pathway, resulting in the game being able to be completed.
* Adding a check to ensure that fastType and instaType could not be selected at the same time. If instaType is enabled, fastType will be disabled and vice versa.
* Fixed the ability to edit keyboard binds in-game, by making the variables that they were supposed to change editable.
* Added the ability to edit the speed of fastType.

This final fix, the speed of fastType being unchangeable, created another bug where the speed was reset when the game restarts. This was fixed by implementing a way to force the game to save the keyboard binds manually whenever the user wants, as long as they are in the menu screen. The keyboard binds also save on quit, but do not save on crash. This is another bug that needs to be fixed, but I have not been able to find a solution as of yet. This bug fix also meant that I had to update the configuration file to add the new options.

The tests that I performed were extensive, and can be seen in the table below.



# Evaluation

Now that I have finished with the programming of the project and testing the product, I believe that I can evaluate the project fully. I believe that the project went smoothly. I think that I did not suffer too much scope creep; my end product has most of the features that were intended at the start of the project, even though I changed product design and programming language at the start of the project, meaning that some of the things within the original brief had to be edited to fit the new format.

I believe that I could have made the product prettier, but given that it was a prototype I think that the outcome was quite good. It emulates the style of old games that were produced in the 1980s, especially the text type which I think is one of the best parts of the program. If I were to revisit this project, then I would work on understanding the Android framework better in order to stick to the original brief.

# Code

*Please note that this is a heavily simplified version with not all of the brief’s specifications in it.*

﻿using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.IO;

using System.Configuration;

namespace StoryTellerTextBasedAdventureGame {

public class Config {

public static bool debug = true;

public static FileStream Q = new FileStream("config.txt", FileMode.OpenOrCreate, FileAccess.Write);

public static char gamePlay\_next\_upper = ConfigurationSettings.AppSettings.Get(

#region URL

"C://Users/natha\_000/Documents/Visual Studio 2015/Projects/StoryTellerTextBasedAdventureGame/StoryTellerTextBasedAdventureGame/" +

#endregion

"App.config/config/gamePlay/next\_upper").ToCharArray()[1];

//public static char gamePlay\_next\_upper = 'C';

public static ConsoleKey gamePlay\_back = ConsoleKey.Backspace;

public static ConsoleKey gamePlay\_exit = ConsoleKey.Escape;

public static bool text\_textType = true;

public static int text\_textFreq (int value) {

//int text\_textFreq = 0; // DEBUG, COMMENT OUT IF NEEDED.

int text\_textFreq = 45000; //LIVE, COMMENT OUT IF NEEDED.

value = text\_textFreq;

return value;

}

public static ConsoleColor foregroundColor = ConsoleColor.Gray;

public static ConsoleColor backgroundColor = ConsoleColor.Black;

public static int percentLost = 10;

}

class Program {

static bool noType = false;

static bool gameRunning = false;

#region Constants

const int ms = 1000;

#endregion

#region Settings

bool settings\_text\_textType = Config.text\_textType;

static int settings\_text\_textFreq = Config.text\_textFreq / ms; // In Seconds

static char settings\_gamePlay\_next\_upper = Config.gamePlay\_next\_upper;

char settings\_gamePlay\_next\_lower = char.ToLower(settings\_gamePlay\_next\_upper);

static ConsoleKey settings\_gamePlay\_back = Config.gamePlay\_back;

static ConsoleKey settings\_gamePlay\_exit = Config.gamePlay\_exit;

#endregion

#region Strings

#region GameText

string gameText\_next = "\n\nPress " + settings\_gamePlay\_next\_upper + " to continue. ";

string gameText\_back = "\nPress " + settings\_gamePlay\_back + " to go back. ";

string gameText\_exit = "\nPress " + settings\_gamePlay\_exit + " to quit.";

#region ImportantWords

static string gameText\_importantWords\_game = "StoryTeller";

static string gameText\_importantWords\_gameTitle = gameText\_importantWords\_game + " - The Virtual Gamebook";

static string gameText\_importantWords\_coins = " Coins"

#endregion

#region Welcome

string gameText\_welcomeText\_Welcome = "Hello, and welcome to StoryTeller \nIn this game, you make the decisions and the story builds around your actions. \nWhen prompted, type where you want to go or what you want to do. \nYou shall be prompted with what options you can do. \n\nThe following is an example: \nYou enter a room. You can go 'F'(ORWARDS), 'B'(ACKWARDS), 'L'(EFT), 'R'(IGHT). \nYour choice must match the options given, but does not have to be case sensitive. \nDo not attempt to complete the example; nothing will happen.";

#endregion

#region Menu

string gameText\_menuText\_Main = "Main Menu \n1) Play \n2) Statistics \n3) Achievements \n4) Credits ";

string gameText\_menuText\_Play = "Play \nOh, hi there " + Environment.UserName + ". Pick an option, please.\n1) Start A New Game ";

string gameText\_menuText\_Stats = "Statistics \nTHINGS GO HERE LATER FIGURE IT OUT LATER.";

string gameText\_menuText\_Achievements = "Achievements \nTHINGS GO HERE LATER FIGURE IT OUT LATER.";

string gameText\_menuText\_Credits = "Credits \n" + credits\_title\_leadDeveloper + ": " + credits\_name\_nathanWindisch + "\n" + credits\_title\_writer + ": " + credits\_name\_oliEastGreenCox + "\n" + credits\_title\_writer + ": " + credits\_name\_nathanWindisch + "\n" + credits\_title\_tester + ": " + credits\_name\_jamesCox + " " + credits\_notes\_norelation;

string gameText\_menuText\_PressNumberToSelectItem = "FIX THIS LATER";

#region Play Menu

string gameMenu\_menuText\_playMenu\_resumeGame = "\n2) Resume Previous Game";

#endregion

#endregion

#region StoryText

#endregion

#region Credits

#region Titles

static string credits\_title\_leadDeveloper = "Lead Developer";

static string credits\_title\_writer = "Writer";

static string credits\_title\_tester = "Tester";

#endregion

#region People

static string credits\_name\_nathanWindisch = "Nathan Windisch";

static string credits\_name\_oliEastGreenCox = "Oli Eastgreen Cox";

static string credits\_name\_jamesCox = "James Cox";

#endregion

#region Notes

static string credits\_notes\_norelation = "(No relation)";

#endregion

#endregion

#endregion

#endregion

int coins = 0;

static void Main (string [] args) {

Console.Clear();

new Program().run();

}

void run () {

Console.Title = gameText\_importantWords\_gameTitle;

Console.SetWindowSize(100, Console.WindowHeight);

Console.ForegroundColor = Config.foregroundColor;

Console.BackgroundColor = Config.backgroundColor;

while (!Console.KeyAvailable) {

Console.Clear();

//Console.WriteLine(random (0, 5));

type(gameText\_welcomeText\_Welcome);

type(gameText\_next);

noType = true;

ConsoleKeyInfo next = Console.ReadKey();

if (next.KeyChar == settings\_gamePlay\_next\_lower || next.KeyChar == settings\_gamePlay\_next\_upper) {

Console.Clear();

MainMenu();

} if (next.Key == ConsoleKey.Escape) {

Environment.Exit(0);

} else {

Console.Clear();

run();

}

Console.ReadLine();

}

Console.ReadKey();

}

void MainMenu() {

//prepend();

type(gameText\_menuText\_Main);

append();

ConsoleKeyInfo next = Console.ReadKey();

if (next.KeyChar == '1') {

prepend();

PlayMenu();

}

if (next.KeyChar == '2') {

prepend();

StatisticsMenu();

}

if (next.KeyChar == '3') {

prepend();

AchievementsMenu();

}

if (next.KeyChar == '4') {

prepend();

CreditsMenu();

}

if (next.Key == settings\_gamePlay\_back) {

prepend();

MainMenu();

} if (next.Key == ConsoleKey.Escape) {

Environment.Exit(0);

} else {

//prepend();

MainMenu();

}

}

void PlayMenu() {

type(gameText\_menuText\_Play);

if (gameRunning) {

type(gameMenu\_menuText\_playMenu\_resumeGame);

}

ConsoleKeyInfo next = Console.ReadKey();

if (next.KeyChar == '1') {

prepend();

game();

}

if (gameRunning) {

if (next.KeyChar == '2') {

prepend();

resume();

}

}

if (next.Key == ConsoleKey.Backspace) {

prepend();

MainMenu();

} if (next.Key == ConsoleKey.Escape) {

Environment.Exit(0);

} else {

prepend();

PlayMenu();

}

}

void StatisticsMenu() {

//type(gameText\_menuText\_Stats);

type(gameText\_next);

wip();

MainMenu();

append();

ConsoleKeyInfo next = Console.ReadKey();

if (next.Key == ConsoleKey.Backspace) {

prepend();

MainMenu();

} if (next.Key == ConsoleKey.Escape) {

Environment.Exit(0);

} else {

prepend();

StatisticsMenu();

}

}

void AchievementsMenu() {

//type(gameText\_menuText\_Achievements);

wip();

MainMenu();

append();

ConsoleKeyInfo next = Console.ReadKey();

if (next.Key == ConsoleKey.Backspace) {

prepend();

MainMenu();

} if (next.Key == ConsoleKey.Escape) {

Environment.Exit(0);

} else {

prepend();

StatisticsMenu();

}

}

void CreditsMenu() {

type(gameText\_menuText\_Credits);

append();

ConsoleKeyInfo next = Console.ReadKey();

if (next.Key == ConsoleKey.Backspace) {

prepend();

MainMenu();

} if (next.Key == ConsoleKey.Escape) {

Environment.Exit(0);

} else {

prepend();

StatisticsMenu();

}

}

void game() {

warning("PLEASE NOTE THAT THIS IS A DEEP ALPHA VERSION OF THE GAME; IT WILL NOT BE THIS EASY.");

type("You wake up in a room. Do you go... \n1) Left \n2)Right");

ConsoleKeyInfo next = Console.ReadKey();

if (next.KeyChar == '1') {

type("You went left and DIED. \nPress " + gamePlay\_next\_upper + " to continue.");

ConsoleKeyInfo next2 = Console.ReadKey();

if (next2 == gamePlay\_next || gamePlay\_next\_upper) {

lose();

}

}

if (next.KeyChar == '2') {

type("You went right and WON. \nPress " + gamePlay\_next\_upper + " to continue.");

ConsoleKeyInfo next2 = Console.ReadKey();

if (next2 == gamePlay\_next || gamePlay\_next\_upper) {

win();

}

}

}

void win() {

type("Congratulations, you have won! Enjoy your some " + gameText\_importantWords\_coins + "!");

success("(+100 coins)")

coins = coins + 100;

}

void lose() {

type("You died! You lost " + Config.percentLost + "% of your " + gameText\_importantWords\_coins + "!");

int lostCoins = (coins \* 100) / Config.percentLost;

coins = coins - lostCoins;

}

void resume() {

wip();

MainMenu();

}

void random(int i, int j, int x) {

Random random = new Random();

x = random.Next(i, j);

}

void type(string value) {

for (int i = 0; i < value.Length; ++i) {

Console.Write(value.ToCharArray()[i]);

wait(settings\_text\_textFreq);

}

}

void wip() {

warning("This is a Work In Progress, sending you to the Main Menu instead.\n\n");

}

void success(string value) {

Console.ForegroundColor = ConsoleColor.Lime;

type(value);

Console.ForegroundColor = Config.foregroundColor;

}

void warning(string value) {

Console.ForegroundColor = ConsoleColor.Yellow;

type(value);

Console.ForegroundColor = Config.foregroundColor;

}

void error(string value) {

Console.ForegroundColor = ConsoleColor.Red;

type(value);

Console.ForegroundColor = Config.foregroundColor;

}

void wait(int i) {

System.Threading.Thread.Sleep(i);

}

void prepend() {

Console.Clear();

}

void append() {

type("\n");

type(gameText\_back);

type(gameText\_exit);

}

}

}

# Bibliography

Figure I – Dark Theme: <http://play.google.com/>

Figure II – Micro-transactions: <http://giantbomb.com/>

Figure III – Colossal Cave Adventure: <http://pcworld.com/>

Figure IV – XML: <http://sqlauthority.com/>

Figure V – Data Type: <http://redbubble.com/>

Figure VI – Save Data UML Diagram: <http://yuml.me/>

Figure VII – Brief Flow Chart: <http://draw.io/>