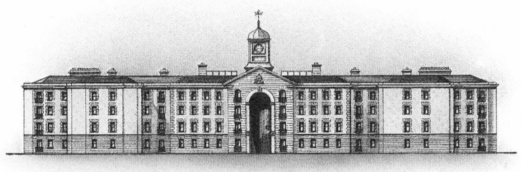


GRIFFITH COLLEGE DUBLIN

**Student name:**

**Student number:**

**Faculty:** **Computing Science**

**Course:** **BSc / HCC / Other** **Stage/year:** **1**

**Subject:** **Business Information Systems**

**Study Mode:** Full time _____ Part-time **X**

**Lecturer Name:** **Eoin Carroll**

**Assignment Title:**

**No. of pages:**

**Disk included?** Yes / No

**Uploaded to Moodle?** Yes / No

**Additional Information:** (ie. number of pieces submitted, size of assignment, A2, A3 etc)

Date due: **25/03/2018**

Date submitted: **25/03/2018**

**Plagiarism disclaimer:**

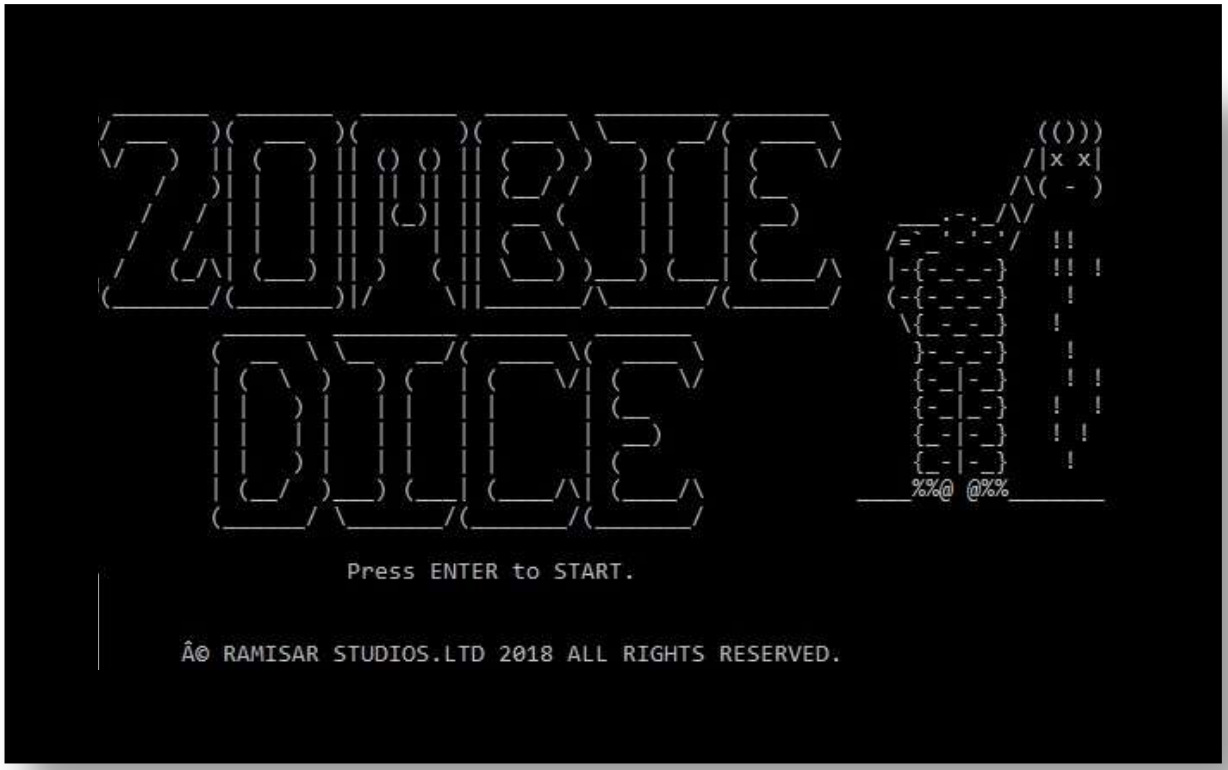
*I understand that plagiarism is a serious offence and have read and understood the college policy on plagiarism. I also understand that I may receive a mark of zero if I have not identified and properly attributed sources which have been used, referred to, or have in any way influenced the preparation of this assignment, or if I have knowingly allowed others to plagiarise my work in this way.*

*I hereby certify that this assignment is my own work, based on my personal study and/or research, and that I have acknowledged all material and sources used in its preparation. I also certify that the assignment has not previously been submitted for assessment and that I have not copied in part or whole or otherwise plagiarised the work of anyone else, including other students.*

**Signed:** **Date: 25/03/2018**

**Please note: Students MUST retain a hard / soft copy of ALL assignments**

# Design Document of ZombieDice



# Contents

# Introduction

The goal of this document it's to present in a simple, short and clear way the main idea that drove me to make some decisions while coding my game. I will present some problems that I had during the project; some codes that I had to implement to get what I was originally planning and some concepts that I will need to improve in my next projects.

The ZombieDice Game it's a base turn dices game which the player assumes a zombie. The goal its to get 13 victims (represented by "Brains"), pursue victims who had escaped (represented by "Footprints") and avoid get hurt (represented by "Shoots").

# The Idea

## Step by Step

To manage to finish my game, I divided the project in different steps which I believe that was the key for my conclusion. Starting from something small and simple and advancing during the project, helped me to go deep in the main idea of the game and even when I had to implement it from a single player game to a multiplayer game, wasn't so difficulty because the basic functions were already working.

The division was made in the following stages:

### Dices

- Having a dice game, the first step for me was to create the dice's mechanism. Random faces, rolling it and printing the results were my first steps in the early stages.

### Main Menu

- After concluded the previous step, I decided to organize the main ideas for my project with the main menu.
    - NewGame would represent the start of my program.
    - Credits would be there just for design.
    - Quit game would finish the program and stop the looping.

### Eliminating the Simple Tasks

- Before focussing my energy in the main idea of the game (game & rules), I decided to eliminate the simple task, so I could dedicate more time for the game itself.
    - So, I decided to create the design for the "Press Start" stage.
    - The Credits menu was created with a Lorem Ipsum text.
    - Quit option was concluded.
    - I collected all the ASCII images that I would use in my project, so I could be running the tests and check already how it would work while playing.
- The methods were organized into different functions. I tried to keep each different feature into a method, so the code could be organized and clear for everyone. The only problem was the main method which I left too short, which I think it's incorrect.

## Footprint

- Took me some time to figure out how I would manage to collect the Footprint output and work with it.

## The Game

- First, I started to focus in small functions such as:
  - When collecting Brains, the player would score.
  - Passing turn the player would save the score.
  - Losing lives would make the player lose the scores for this turn.
  - "Getting 13 Brains" the player wins.
  - After win, player would have option to start the game again.
  - In this stage I was considering just a single player.
- With game working with all the main features for a single player, I started to move forward.
  - I started collecting the total of players and names.
  - Using the player's list, I decided to develop the turns idea
  - My last step was to synchronize the players and their scores.
  - This stage was the longest one and took me some time. Not just because the code itself, but I had to implement the previous steps to work with multiplayers.

## Final Stage

- Playing the game, I managed to improve a lot of bugs and concepts.

# Main Menu

## Design

The design of my game is totally based in old games. Which the player used to have to press any key to start the game.

Some examples:


## ‘1’ and ‘2’

The original idea was to collect the keys ‘Y’ and ‘N’ as inputs from the user. But after some time running tests, I have decided to replace it by ‘1’ and ‘2’, which it’s much quick and easier for the player to use.

The fact that the keys are just beside each other, makes the game smoothly progress from one stage to another one.

### “Char” Menu

Using all the knowledge that I have so far and avoiding codes from forums, I decided to collect the user’s inputs using Char variables instead of Integers. The main reason, is because I was trying to avoid the game to clash every time that the user input another type of data. Example: if I was collecting integers, the game was going to clash when receiving letters/symbols.

Using char variables, I have total control to get the desirable data and deal with another character different from ‘1’ and ‘2’.

# Players

### ArrayList

To avoid a limit of players and to work better with list, I have decided to use ArrayLists. I was trying to use Arrays, but took me a lot of tests and I couldn’t figure out some way to collect the total of players from the user and use it as my limit.

So the easiest way by the end was to study a bit about ArrayLists and use it. The ArrayLists provide a flexible list which can be extended by the user and works similar to Arrays in the other features.

### Turns

Using the ArrayLists, I collect the player’s information from the user; I add it to the list, them using the player’s index position, I basically control all my other functions. That’s was perfect when dealing with turns and scores.

Dealing with turns I just had to get the total of players – 1, which in this case “-1” represents the total of positions that I have in the list subtracting the “0” position. This way, I would link the player position “0” with the turn equals to ‘0’ and so on.

For example: a list with 5 players length will result in the positions (0,1,2,3,4). So without subtracting “-1” from the total, I would have the integer 5 which would give (0,1,2,3,4,5) or in other words, a $6^{th}$ player turn which is incorrect.

### Scores

To deal with the scores, I decided to use a simple Array. Different from the Players situation, which I had difficulty to set the limit collecting it from the user, I linked the length of the Array with my turn variable. This way the player's position '0' in my ArrayList, would be represented by the turn '0' which would have your score saved in the Array over the first position: '0'. Creating so a looping, I managed to synchronize these information, managing to loop between the lists and linking them together.

# Dice

### Position, colours and faces

I divided my dice into three positions (1,2 and 3) which would represent the order that the dice was rolling. Them, using a number between 1 and 13, I divided the dice into colours being 1 till 6 the green dice; 7 till 10 the yellow dice and finally the 10 till 13 the red dice.

Finally, the faces where created in a different method, collecting it depending on the dice's colours.

For example: A green dice should have 3 brains, 2 footprints and 1 shoot.

In this stage I decided to use the faces in a Array as Strings because I would like to make the code easier to read and understand. I could have done it using numbers for example, but I think that the Array was a better approach.

### FootPrint

The Footprint took me a lot of time, especially because I didn't have any idea how I would work with it. First, I thought that would be something easy, just collecting, storing and using it as necessary. But then I realized that I needed to store them by colors, quantity and how this would affect the whole dice function. Affecting since the player's turn till the quantity of dice per turn.

I solve this problem using a piece of paper and pen. I drew the looping and then I realized some questions such as "what's the foot's color?", "how many I have this turn?", etc. Finally, I created:

- A variable for the colors: green, yellow and red.
  - The quantity of each one per turn.
- The quantity of footprints in a turn.

        - The total of footprints so I could deducted it from the total of dice per turn.
    - An exclusive looping just to deal with the footprints in the begging of the turn.
        - This way, the total of footprints from the previous turn would be deducted.
    - And the total of dice per turn would be respected (three). Avoiding the game to create more dice per turn.
        - First time when I coded I was having the three dices per turn plus the footprints from the previous turn resulting in till six dice in a single play. Which obviously was incorrect.

# ASCII Arts

## Design

This was actually the first idea when I started the project. Just because when I was young, I used to upload these arts in forums, that back in the 90's, didn't allow many pictures or emojis. Also, some of them were used as signs for your profile, so every time that you added a new message to a forum, by the end of the message it would contain your signature which normally would be an ASCII Art.