Assignment 2

Zombie Dice

This project will allow you to implement a computer game, with some simple graphics. It is a "push your luck" game involving zombies eating brains, where you will implement some simple artificial intelligence for a computer player.

This project is worth 7.5% of your final grade. You must do this project on your own. The submission deadline is the last teaching day of semester, Friday 11th October 2024 at 5pm. Your submission will be marked out of 12 in the lab, then your C# code will be marked out of 8 after you submit your program.

Project Specification:

You have been asked to write a computer version of the "Zombie Dice" game, by Steve Jackson. The game involves rolling dice, where you want to collect (eat) as many brains as you can, without getting three shotguns. As the box says: "Eat brains. Don't get shotgunned."

Your program should implement a 2-player version of the game where the user plays against the computer, or another player.

Overview[1]

In Zombie Dice, you are a zombie. You want *braaains* – more brains than any of your zombie buddies. The 13 custom dice are your victims. Push your luck to eat their brains, but stop rolling before the shotgun blasts end your turn! Whoever collects 13 brains first wins. Each game takes 10 to 20 minutes and can be taught in a single round.

Each turn, you randomly take three dice from the box and roll them. A brain symbol is worth one point at the end of the round, while footsteps allow you to reroll this particular dice. Shotgun blasts on the other hand are rather bad, cause if you collect three shotgun blasts during your turn, it is over for you and you get no points. After rolling three dice, you may decide if you want to score your current brain collection or if you want to push your luck by grabbing new dice so you have three again and roll once more.

[1] Rules and description taken from https://boardgamegeek.com/boardgame/62871/zombie-dice

Game rules:

This game includes these rules, 13 dice, and a cup to hold them. You'll need some way to keep score. Two or more can play.

The first player is the one who won the last game, or the one who can say "*Braaaaains!*" with the most feeling.

Mmm! Brainnns!

On your turn, shake the cup, take three dice from it without looking, and roll them. Each one is a human victim. The red dice are the toughest. Green are easiest, and yellow are medium tough.

The dice have three symbols:

- **Brain** – you ate your victim's brain. Set your Brain dice to your left.
- **Shotgun** – he fought back! Set your Shotgun dice to your right.
- **Footprints** – your victim escaped. Keep your Feet dice in front of you. If you choose to roll again, you will re-roll these dice, along with enough new ones to bring the total to three.

If you rolled three shotguns, your turn is over. Otherwise, you can choose to stop and score, or continue.

If you decide to **stop**, score 1 for each Brain you have, and put all the dice back into the cup. It's the next player's turn.

If you choose to **keep going**, leave all your Feet on the table. Unless all three of your dice are Feet, take enough random new dice from the cup to total three, and roll again. Whenever you roll, you will roll three dice at a time.

After you take new dice, you can't decide to stop ... you have to roll.

Set aside Brains and Shotguns as above. If you are up to 3 Shotguns, your turn is over and you score **nothing**. Otherwise, you can stop and score, or take another roll …

Brrrains?

If you don't have three dice left in the cup, make a note of how many Brains you have and put them all in the cup (keep the Shotguns in front of you). Then continue.

BRAAAINS!!!

Play until someone reaches 13 Brains. Then finish the round. Whoever has the most Brains at the end of that round is the winner. If there's a tie, the leaders (only) play a tiebreaker round.

Dice colours

For the standard version of the game, there are three types of dice.

- The six *green* dice are easy and have 3 *Brains*, 1 *Shotgun*, and 2 *Feet* as their possible outcomes.

- The four *yellow* dice are medium and have 2 *Brains*, 2 *Shotguns*, and 2 *Feet* as their possible outcomes.

- The three *red* dice are the toughest and have 1 *Brain*, 3 *Shotguns*, and 2 *Feet* as their possible outcomes.

Extension

Zombie Dice version 2 adds two more kinds of dice: the zombie-fighting Hollywood heroes called the *Hunk* and the *Hottie*. The Hunk is a black die with white icons. The Hunk has 2 *Feet*, 2 *Shotguns*, 1 *Double Shotgun*, and 1 *Double Brain*. The *Hottie* is a black die with pink icons. The Hottie has 3 *Feet* (with high heels!), 2 *Shotguns*, and 1 *Brain*. Remove two yellow dice from the cup, and drop in the *Hunk* and *Hottie*.

The Heroes can rescue each other from the zombies. If the *Hunk* rolls *Shotguns* and the *Hottie* is in the collected *Brains* (or came up *Brain* on the same roll), the *Hottie* is rescued. That is, the *Hottie* is removed from the collected *Brains* and goes back into the cup. The *Hottie* can rescue the *Hunk* in the same way.

Change your program to support the play of Zombie Dice version 2.

Robot player

The game can be written where humans control both players, or you can write a robot player to play against.

At each turn the robot player needs to roll a set of three dice. The robot then needs to decide whether to stop or continue rolling more dice.

A purely random robot player would not be that interesting to play against. So, your robot player must try to have some artificial intelligence, making its choice based on the values rolled, how close the robot and player are to winning, and the score difference between the robot and the player. You may want to let the user choose from robots that play in different styles.

Tasks

Write a program that satisfies the specifications described above. It should make use of all of the programming features that you have learnt so far, including using classes and methods to structure your program and XML documentation where appropriate.

The project requires you to both get it verified in the lab and to hand in the source code. Half of the grade will be based on how the program runs and half on the object design, structure, code style and documentation.

Suggested Steps

We suggest that you build a simplified version of the game first, and then add in more features as time permits. Concentrate on getting the objects and simple turn play right before adding fancy graphics and controls.

Start with only one type of dice. This means that filling the cup with 3 dice is simplified and you can concentrate on the rolling and outcomes.

You may want to generate the dice colours randomly to start with, and then add code that keeps track of dice used to get the correct number of each coloured dice.

For testing purposes, you probably want to be able to specify how many points are in the game, so a game can be shorter than 13 brains.

It is often easier to control both players, adding a computer player once you have a basic game going. Once you have a stupid computer player, then add more intelligence and heuristics as you have time.

COMPX102-24B

Assignment 2 Hand-in due 5pm 11th October 2024

Compress (Zip) the Visual Studio folder with your program code and submit it via Moodle.

Student Declaration of Originality

I declare that the program which I have had verified and submitted in Moodle is entirely my own work. I have not worked together with any other people. I have suitably acknowledged (referenced) any parts of other programs that I have used. I understand that if I have breached the above conditions, I will be sent to the University Disciplinary Committee.

Name: Aidan Pine

ID Number: 1655204

Signed: [illegible]

Date: 11th October 2024

Functionality and Usability (demonstrated in the lab) _____ /12 marks

	Minimal or none	Partial, simple or glitchy	Works
Shuffle the dice in the cup on each turn	0	½	1
Player can roll dice to generate outcomes	0	½	1
Turn is over if three shotguns collected	0	½	1
Player can collect feet and roll 3 dice again	0	½	1
Brains added to score when player ends turn	0	½	1
Reuse Brains dice when < 3 dice left in cup	0	½	1
Game continues turns until game won	0	½	1
Possible outcomes for each colour implemented	0	½	1
Enum used to represent the dice outcomes	0	½	1
Hunk or Hottie dice implemented	0	½	1
Second player implemented	0	½	1

	Poor	Hard/Confusing	Easy
Usability: layout, choice of controls, feedback, etc	0	½	1

Bonus mark: Two or more delegates defined and used correctly _____ /+1 mark
Bonus mark: Robot players in game _____ /+1 mark

Coding Style (marked by tutor after zip submitted) _____ /8 marks

	Poor	Average	Good	Excellent
Object design, inheritance, composition, scope	0	1	2	3
Code style, user methods, code structure	0	1	2	3
Documentation: XML and inline comments	0	1	2	2

Total: _____ /20 marks