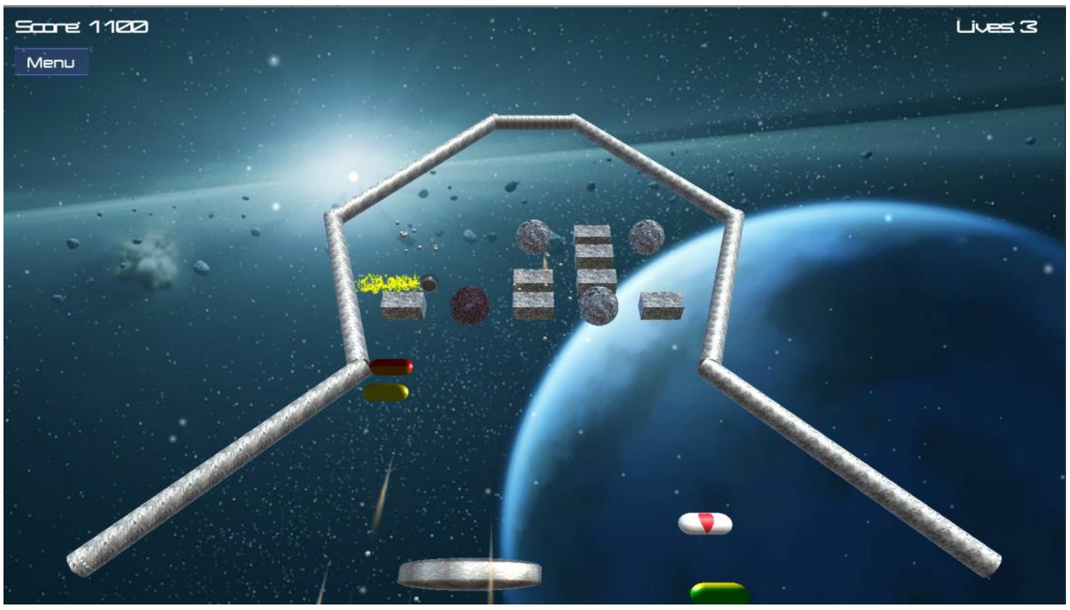
COMP30019 Project 2 Report

Meteor Buster

Michael Marshall	–	758622
Myles Adams	–	761125

Tutorial: Monday 3:15pm

A brief explanation of the game:

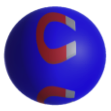


Meteor Buster is a simple 'paddle and ball' game. The aim of the game is to get the highest score possible before you die by running out of lives and to complete all three stages of the game.

The player has the responsibility of the paddle at the bottom of the screen. They need to try and not let the ball (read: asteroid) fall beneath the paddle. This game has a main menu and three different levels, which increase in difficulty as the player progresses. The main menu contains the option to adjust the volume and perhaps more importantly adjust the difficulty of the game (easy, medium or hard). The difficulty adjusts the speed of the ball in the game.

All objects have rigid body properties, so the ball ricochets or bounces off any object in its path. Each different block (read: meteor) has a certain amount of health and a specific score associated once it is destroyed. The rectangular blocks require one hit to crumble, the diamonds require two hits and the circles require three hits.

Additionally, the blocks may randomly drop one of five different power-ups that alter the gameplay, which are activated when they land on the paddle. Besides the 'shrink' power-up, all other power-ups are advantageous for the player and should be targeted.

Type	Image	Description
Magnet		Makes the ball stick to the paddle, allowing the player to choose the balls release point
Life		Gives the player an additional life
Powerball		Alters the property of the ball, allowing it to destroy any object instantly without deflection
Shrink		Reduces the size of the paddle
Enlarge		Increases the size of the paddle

How to use it (especially the user interface aspects):

Meteor Buster is a very straightforward game to play by design. Only three keys are required to play the game: spacebar, left arrow and right arrow. The spacebar key is used to release the ball from the paddle at the start of the level, after a death and when using the magnet power-up. The left and right arrow keys are used to control the paddle so that the ball stays in the arena.

From the main menu, the player can personalize their audio and difficulty preferences before playing the game.

How have you modeled the objects and the entities:

All the objects in this game take advantage of Unity's in-built 3D game objects. The boundary walls are cylinders, the paddle and power-ups are capsules, the ball is a sphere and the blocks are either a cube or sphere game object.

This was specifically designed so that the game would look authentic and real. People who have played similar games would understand the game layout and feel comfortable with the layout. We designed and textured the objects such that the game suited the space theme without compromising on style.

Meteor Buster incorporates five different particle effects:

1. The space debris background particle effect
2. The ball's trail particle effect
3. The paddle's death particle effect
4. The destroyed bricks rubble particle effect
5. The pulsing particle effect under the paddle

Whilst the background particle effect was taken from the Unity Store, the other four were specifically created in order to improve stimulation and interaction with the player.

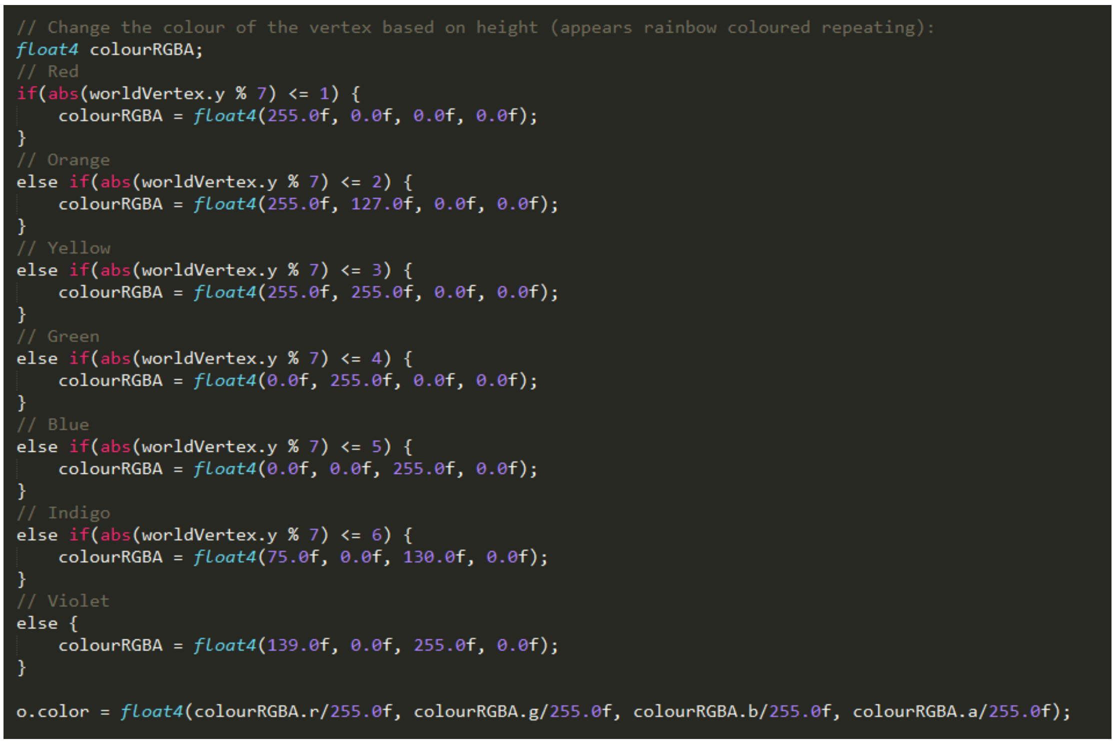
How you handled the graphics pipeline and camera motion

To handle the graphics pipeline we used the standard Post Processing Stack from the Unity Store in conjunction with Unity's inbuilt features. This asset combines a complete set of image effects into a single post-process pipeline, allowing the combination of many effects to create the final display. In particular, we used the bloom, motion-blur and anti-aliasing effects in our game, making our game more visually appealing.

Since this game is an 'arcade' style game, the camera is in a fixed position for the entirety of the game. The camera is position at a 50° slope relative to the walls in order to provide a more stimulating viewing angle.

Descriptions of how the shaders work

We used the Phong shader provided from the tutorials on the ball and power up when in 'power-ball' mode, with a few modifications. In order to create the colour-changing effect, we altered the shader to produce different colours based on the object's Y-axis height. This is akin to gaining invincibility in Mario Kart when consuming a 'star' powerup. Alongside the Phong component, the shader uses modular mathematics to define a repeating pattern of RGB values, which are assigned to the colour component of each vertex during the vertex shader, as shown:

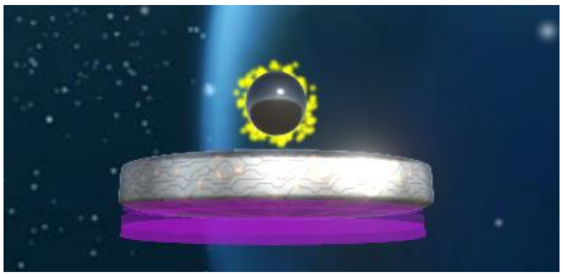


```
// Change the colour of the vertex based on height (appears rainbow coloured repeating):
float4 colourRGBA;
// Red
if(abs(worldVertex.y % 7) <= 1) {
    colourRGBA = float4(255.0f, 0.0f, 0.0f, 0.0f);
}
// Orange
else if(abs(worldVertex.y % 7) <= 2) {
    colourRGBA = float4(255.0f, 127.0f, 0.0f, 0.0f);
}
// Yellow
else if(abs(worldVertex.y % 7) <= 3) {
    colourRGBA = float4(255.0f, 255.0f, 0.0f, 0.0f);
}
// Green
else if(abs(worldVertex.y % 7) <= 4) {
    colourRGBA = float4(0.0f, 255.0f, 0.0f, 0.0f);
}
// Blue
else if(abs(worldVertex.y % 7) <= 5) {
    colourRGBA = float4(0.0f, 0.0f, 255.0f, 0.0f);
}
// Indigo
else if(abs(worldVertex.y % 7) <= 6) {
    colourRGBA = float4(75.0f, 0.0f, 130.0f, 0.0f);
}
// Violet
else {
    colourRGBA = float4(139.0f, 0.0f, 255.0f, 0.0f);
}

o.color = float4(colourRGBA.r/255.0f, colourRGBA.g/255.0f, colourRGBA.b/255.0f, colourRGBA.a/255.0f);
```

These colours exactly match the traditional RGB values defined for a rainbow, in the correct order (ref: https://simple.wikipedia.org/wiki/Rainbow).

Our other shader is incorporated at the bottom of the paddle (shown in the image below) is attached to a capsule-shaped particle emitter. The 'Phasing Shader' implements the vertex shader, transforms the vertex in world coordinates to camera coordinates and then changes the transparency of the vertex using a time-based sine curve, as well as making the object colour purple. As a result, the shader provides a dynamic visualization whilst adding to the space theme of our game.



Description of the querying and observational methods used:

We conducted a post-talk walkthrough as our observational methods and a questionnaire as our querying technique. The questionnaire was adapted from a System Usability Scale in order to mitigate the issue of not being able to do any probing and to provide quantitative analysis.

For the conduction of the post-talk walkthrough, we tested five people from our computer lab during the tutorial. This demographic has a strong computational understanding and have generally had exposure to playing games in the past.

We interviewed 6 people for our questionnaire, all of which were undergraduate students from Ormond College. None of these people had a strong understanding of game development and had a varying range of ability to play computer games. This was preferable to our asking lab peers as this group is a more realistic and diverse demographic, allowing to test our game with a wider audience.

For the post-task walkthrough, one of our members would ask the participants a series of questions from a pre-defined list while the other member wrote down the participants' answers as they went along. For the questionnaire, we used Survey Monkey and had the participants fill out the questionnaire online immediately after playing the game.

Document the changes made to your game based on the information collected during the evaluation:

Based on the information we received there were several aspects that we needed to improve. One of the criticisms we faced was that the game didn't visually look appealing. To combat this we changed the textures of the objects and included a space-themed background with a particle system for extra aesthetics.

Another criticism we faced was that the game was either too easy or too hard depending on who played the game. As a consequence, we included in the main menu an option to select the game difficulty (i.e. the speed of the ball).

We also found that people would not be eager to continue playing this game repeatedly. To aid further interaction we included a scoring system, added more levels and included randomly generated power-ups to add more variability to the game.

The Survey Monkey results can be found here:
https://www.surveymonkey.com/results/SM-639QJJ5D8/

A statement about any code/APIs you have sourced/used from the internet that is not your own:

To complete this project, we sourced inspiration and understanding from a few YouTube tutorials. It is worth noting that whilst there may be an overlap in code from the tutorials, a lot of logic has been adapted to better suit our game. These videos and tutorials can be found below:

- https://www.youtube.com/watch?v=EN29Vd_X4dY
- https://www.youtube.com/watch?v=MiBtGy0rh00
- https://unity3d.com/learn/tutorials/modules/beginner/live-training-archive/creating-a-breakout-game
- https://unity3d.com/learn/tutorials/topics/user-interface-ui/creating-main-menu

Additionally, we sourced our game music, magnet SFX and power-ball SFX from:

- www.bensound.com
- https://www.youtube.com/watch?v=QEyEs9NqLEQ
- https://www.youtube.com/watch?v=XGRhb4z526Y

We also reused some code created or given to us during tutorials (i.e. Phong shader and particle-outside-of-screen death scripts.)

Finally, the background image and particle system was downloaded from Unity's Asset store at:

- https://www.assetstore.unity3d.com/en/#!/content/25468