Earth 2.0 Defender

A JAVASCRIPT GAME

By Janzeb Masiano (P2430705)
IMAT2015 1920 501 | MULTIMEDIA DEVELOPMENT [DE MONTFORT UNIVERSITY]

Multimedia Development

JavaScript Game Assignment

Author: Janzeb Masiano
P-Number: P2430705

Sunday 3rd Nov 2019

Contents

Earth 2.0 Defender

Background Story for the Game

Set in the year 3019, since the near destruction of Earth and almost complete extinction of the human race by an asteroid storm, humans on Earth came together and sent out explorations to newly discovered planets with the potential to support life. One such Earth was discovered in 3010 and was shortly colonized by a small group of 100 humans. This Earth was named Earth 2.0 (pronounced Earth two point 'O').

The new Earth 2.0 human colony is now under threat of being wiped out by an asteroid shower and only one-man Zeb is there to help stop it destruction.

A storm of asteroids is about to hit the New Earth and wipe out the newly established colony of 100 humans. A squadron of destroyers are on their way from Earth, but even at the their fastest ultra-light speeds they will be over 20 minutes late. By then it may be too late.

You (Zeb) are the only one able to pilot the only destroyer on Earth 2.0 capable of taking on the storm of asteroids. Your job is to destroy as many of the asteroids you can and hope you get enough asteroids before the squadron of destroyers arrives to save the colony of humans on Earth 2.0 being wiped out.

Music and Graphics

The music is a short music loop I created using FL Studio on my Android Phone. The border and screen designs (in the IMAGE folder) were created using PaintShop Pro X8.

Game Play

Browser Compatibility

The game was developed and optimised for playing in Internet Explorer. There were some issues with the game performance in other browsers. I managed to sort out some of the problems but found that if I changed something in one browser it would affect something in another browser. For example, I adjusted the gravity in Internet Explorer, so the rock speed was not too fast, but when I changed to Firefox, it was faster. I tried to get a good balance. I would therefore recommend playing the game in Internet Explorer.

Aim of the game

You have 3 lives, so try not to lose them all or its game over. Dodge the space rocks. If they hit you then you lose a life, unless you invincible (flashing). Simply destroy as many asteroids as you can and get the top score. To start you will not have a weapon to destroy the asteroids. Collect powerups for invincibility to keep you from losing lives, and to change weapons to help you along the way.

Important Note About Starting the Game

When you click the start button the game will begin. At this stage you must ensure that you click anywhere on the screen, away from the start button. This will reposition the pointer off the start button so that when you press spacebar it doesn't click the start button again and restart the game.

Spaceship Controls

The ship can move up, down, left and right. These are controlled by the arrow keys. When you move forward there is an automatic thrust (acceleration). After collecting your first weapon by collecting a powerup ball, you will be able to fire the weapon using the spacebar.

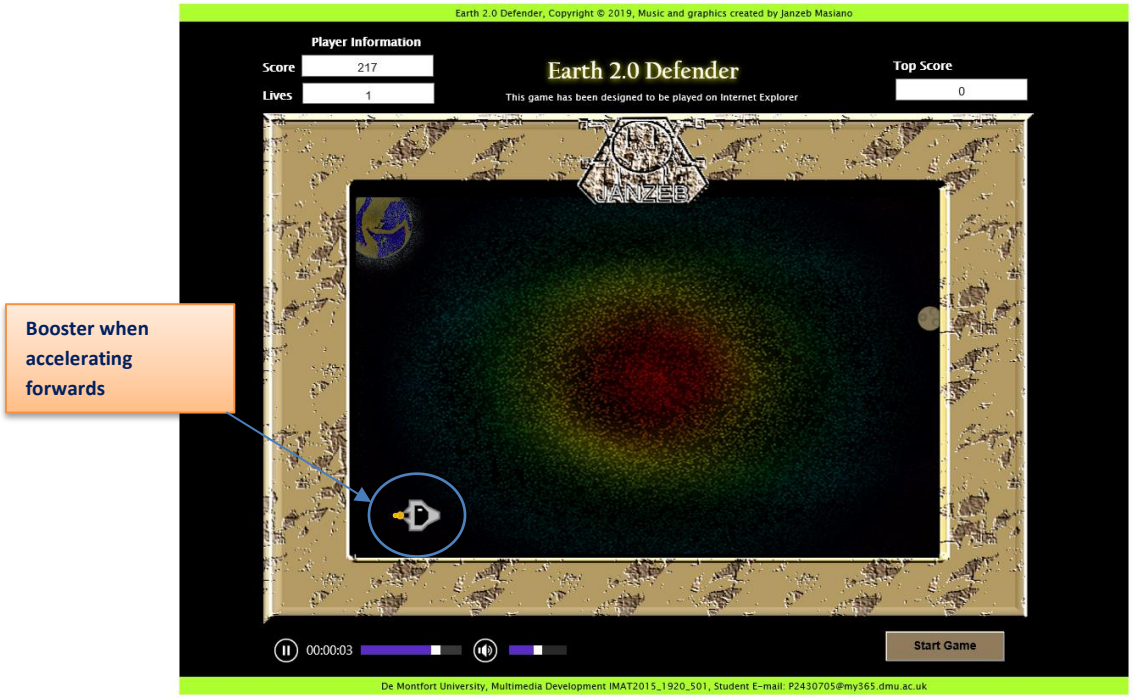
About Collecting the powerup balls

These powerups are different types of weapon. The colour of the powerup ball corresponds to the different weapons. Yellow is Laser, Red is Power Laser, Green is Ray and Blue is the Wave weapon. Each weapon has a different power level. Get a more powerful weapon and you can blow rocks up more quickly. The weaker weapons enable you to get more hits off a rock and so you can get a higher score. The stronger weapons enable you to destroy the space rocks more easily, so fewer hits and less score.

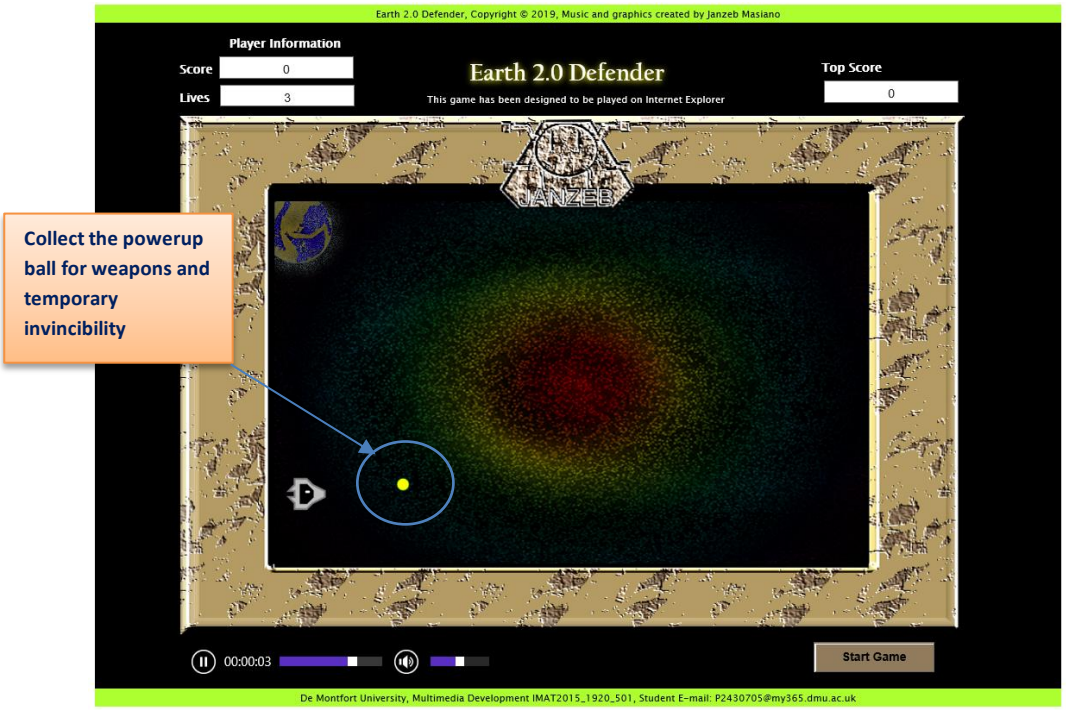
How to Play

(1) Click start button in the bottom right corner to start the game. Restart the game using the same button. **Remember to click on an area other than the start button before resuming the game**. Control the audio using the audio controls. View your score, top score and number of lives

(2) Navigate using the arrow keys. Accelerate forwards (see the booster)



(3) Collect the powerup balls for weapons and invincibility. Different colours for different weapons. Each weapon has a different power level.



Types of weapon

Image for the Laser weapon

Image for the Power Laser weapon

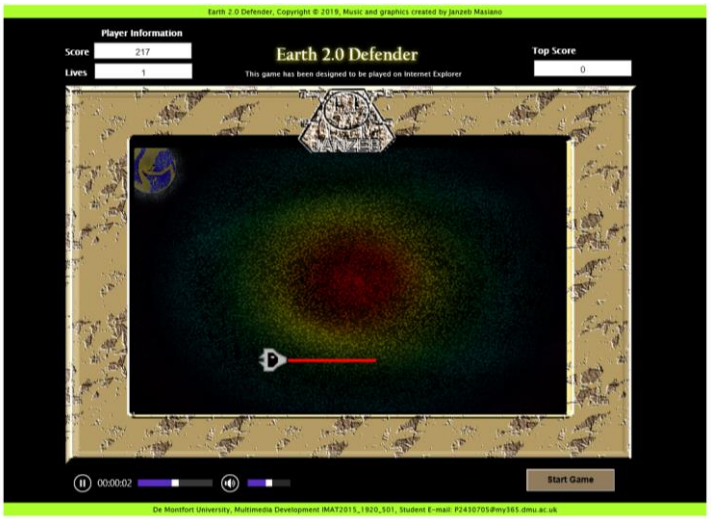
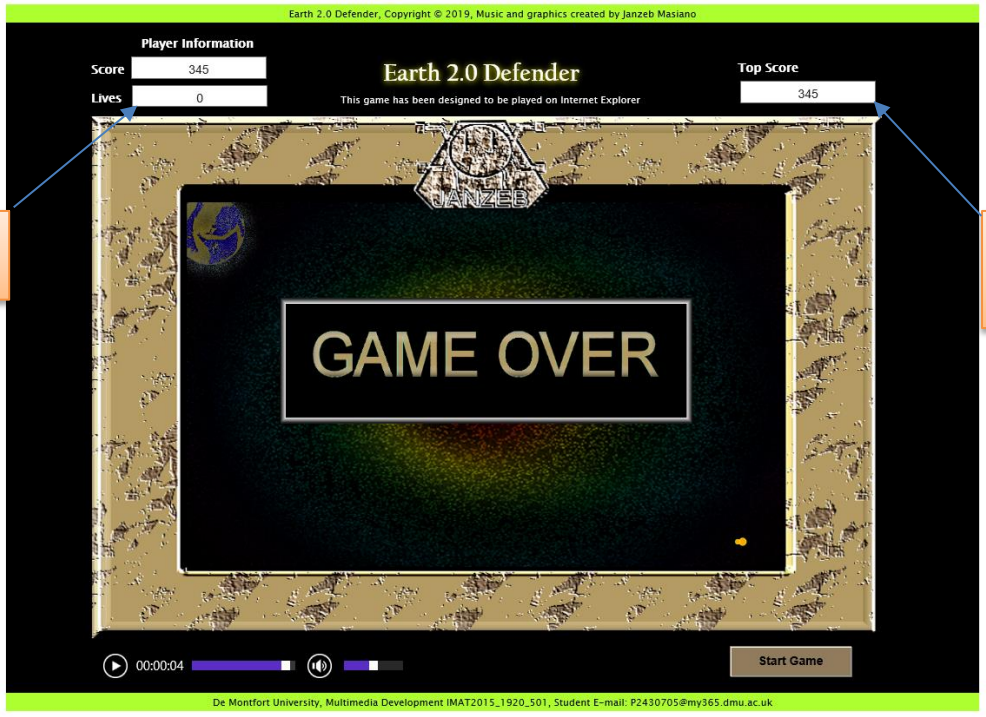


Image for the Ray weapon

Image for the Wave weapon

(4) Don't get hit by the space rocks or you will lose a life. And end up at the start location (top left corner of the screen)

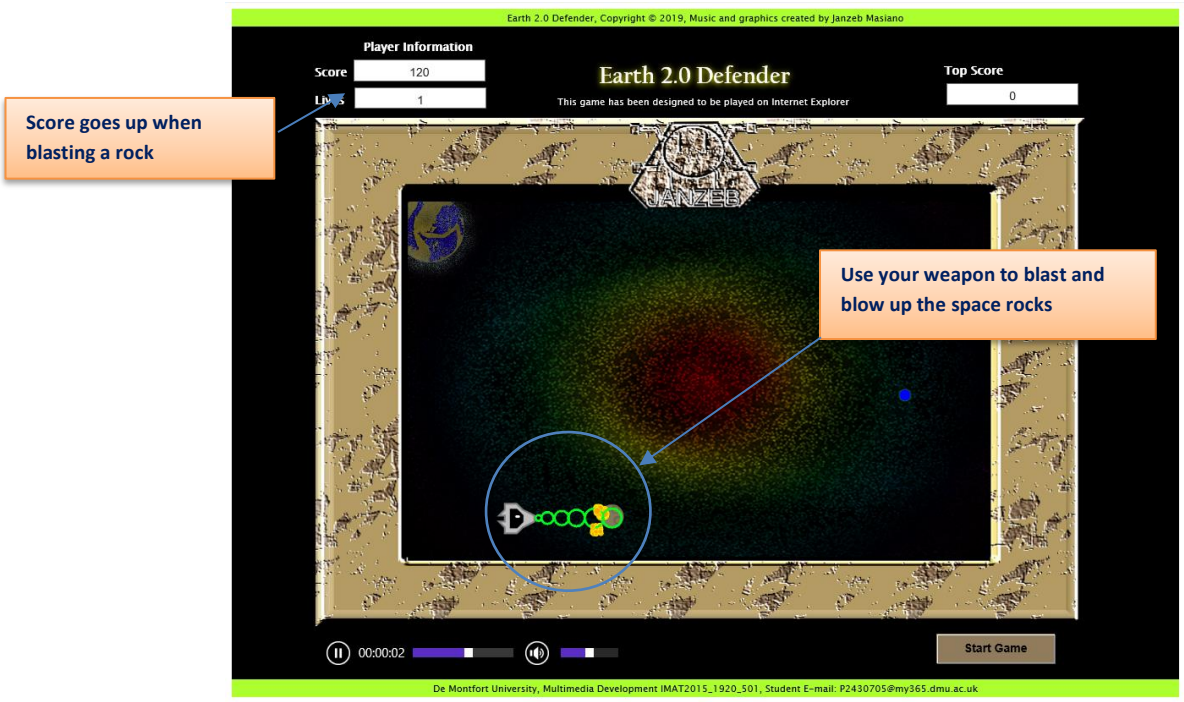
(5) Lose all your lives and its game over. If you get the highest score, then High Score is updated



Lives are 0, so its Game Over

High Score is updated at the end of the game

(6) Use space bar to Blow up the rocks with your weapon and watch your score go up. Hold down the spacebar for continuous fire.



Classes in the Game

Spaceship Class

Sprite design and internal animation

The ship is designed and coloured in using three different sprites. The first is the ship body itself (the drawship body function) the second is the window inside the spaceship (DrawWindow function). The third is for internal animation (ScreenFlash function)

Constant velocity

The space ship is designed to have velocity in all four directions. This is achieved using the velocity variables vx and vy to increment or decrement the x and y co-ordinates.

Acceleration

Acceleration is only in the forward direction (when booster is on) and this is achieved using the Acceleration class by passing in an acceleration object to access the value of the acceleration variable (af) to add to the velocity. Since there is no acceleration in the other directions, the acceleration value can be reset on change of direction by calling the StopForwardAcceleration method. This ensure that the next time we accelerate the acceleration variable starts from zero.

Booster Class

The booster class comprises of the booster draw function containing the sprite for the booster which is just two circles. The booster is drawn on screen at the specified location of the spaceship exhaust. This is calculated from the spaceship private member x and y co-ordinates, passed in using the public member variables. As the booster is only on when the spaceship moves forward, the control for this is the pm_boosterOn Boolean variable.

Acceleration Class

The acceleration class has two functions ForwardAcceleration and StopForwardAcceleration. This are used by the spaceship class to accelerate the spaceship in the forward direction and to stop acceleration in other directions.

Weapon Class

The weapon class comprises of four different sprites for the different weapons. A different weapon is set by passing in a value for pm_weaponType in the collision class, when the spaceship collides with a power ball the colour of the ball equates to the weapon type and this value is passed in to the weapon object to select the weapon. The weapon is fired (drawn on screen) when the weapon object Boolean pm_FireWeapon is set to true.

PowerUp Class

The PowerUp class contains a sprite for the power up ball. The colour is determined randomly and equates to the weapon type set by the integer constant that is randomly generated. The ball movement is controlled by decrementing x, from screen width to zero, making it move right to left across screen. Each time the ball reaches the beginning of the screen it then is reset to a point to a random point to the right, off screen. The screen size is set larger than the screen visible so that the ball doesn't appear on the screen too often.

Hit Class

The hit class has a draw function that draws circles at 2 random locations each time there is a hit. The x and y coordinates passed in are used to calculate the points where the random circles occur. In Internet Explorer, the special effect created is really cool and looks like balls of fire. This is because the hit object is called many times when the weapon makes contact and is continually being hit.

SpaceRock Class

The space rock class contains the draw function which contains the sprite for the rock (a circle), that contains 3 other smaller circles (meant to be creators). The rock is controlled by acceleration due to gravity. The rock is accelerating from left to right of the screen (gravity can come from any direction in space). It does this by using one of Newtons Equations of Motion $x = ut + \frac{1}{2} at^2$ (u is initial velocity, a is acceleration, t is time). As the initial velocity is zero (from rest) this becomes $x = \frac{1}{2} at^2$. Also, since the object is falling, then acceleration due to gravity is negative. The public class giveGravity is used to pass in the Acceleration object, from which the gravity is obtained

The rock is placed off screen at a random y location and begins to accelerate from rest. When it comes on to the screen it is already accelerating. When it gets to the start of the screen (left side) is resets to a random location right side off screen again. The rock is also reset in this wat when the rock is hit 10 times or collides with the ship.

Collision Class

The collision class handles the collisions between rock and ship. It also handles the collision between weapon and rock. The Boolean collision equations are similar in each case. So for example, in order to determine whether there is a collision I have decided to work out firstly when there is no collision for each side of the spaceship with the rock, (the rock will be treated as a box), then by using not (no Collision) we get the logic for a collision. I have chosen this way because it's easier for me to get my head around. If we take the spacerock and spaceship then there is no collision:

(a) SpaceRock.spaceRockYB < Spaceship.spaceshipYT (the rock is above the spaceship)
(b) SpaceRock.spaceRockYT > Spaceship.spaceshipYB (there rock is below spaceship)
(c) SpaceRock.spaceRockXR < Spaceship.spaceshipXL (the rock is to the left of the spaceship)
(d) SpaceRock.spaceRockXL > Spaceship.spaceshipXR (the rock is to the right of the spaceship)

Player Class

The player class contains the score. When the rock is hit the score goes up. But in the game, this happens many hundreds of times, and so the score goes up by thousands really quich. In order to reduce this, I divided the score down by 100 before returning it to be displayed, using the

CalculatedScore class. Also, the Player class contains a the CalculateTopScore function which is used to calculate whether the current score (after dividing by 100) is greater than the top score. If so it will return the new top score. Otherwise it returns the old one.

Bibliography

Brennan E. (2019) *Newton's Laws of Motion and Understanding Force, Mass, Acceleration, Velocity, Friction, Power and Vectors.* [Online]. Available from: https://owlcation.com/stem/Force-Weight-Newtons-Velocity-and-Mass

Tutorials Teacher (no date) *Display Popup Message Box*. [Online]. Available from: https://www.tutorialsteacher.com/javascript/display-popup-message-in-javascript

Stack Overflow (2012) *Playing audio with Javascript?* [Online] Available from: https://stackoverflow.com/questions/9419263/playing-audio-with-javascript

CodeBlocQ (2016) *Two Ways of Creating an Animation Loop in JavaScript.* [Online]. Available from: http://www.codeblocq.com/2016/05/Two-Ways-of-Creating-an-Animation-Loop-in-JavaScript/

Barbour B. (no date). *If Javascript Is Single Threaded, How Is It Asynchronous?* [Online]. Available from: https://dev.to/steelvoltage/if-javascript-is-single-threaded-how-is-it-asynchronous-56gd

Happy Coding (no date) *Collision Detection*. [Online]. Available from: https://happycoding.io/tutorials/processing/collision-detection?sa=X&ved=2ahUKEwijtefGyanlAhVNThUIHRMoBWcQ9QF6BAgLEAs