# **Economic Analysis with Matrices Project Report-ECO2048**

## **Frogger- By F Society:**

## **Alex Lai (6423858),**

## **Rhys Maher (6485242),**

## **Priyank Patel (6484092),**

## **Sonia Patel (6464677),**

## **Kevin Paul (6462881)**

**Motivation:**

Our first meeting as a group together consisted of us agreeing upon a group name, and a project idea. We all brought many different concepts and ideas of the projects we may want to consider, of course one of the proposals was the classic arcade game, Frogger. We all reached a unanimous decision to go with this as we knew it would challenge us and help add to our skill set of coding and working with MATLAB. From there, we researched the game and familiarised ourselves with main concepts of the game, when we next met, we drew up a plan of action, so we had a guideline to follow. (<https://uk.mathworks.com/matlabcentral/fileexchange/52883-frogger-game>)

**Methodology:**

Through the development of the game, we have come up with multiple ways of creating different aspects of the game, through trial and error we have encountered issues and have managed to solve them accordingly. Given below are examples of the problems we faced and how we overcame them.

Frog movement:

Originally, we wanted to make a grid system using a zeros matrix which allowed us to have a moving block. However, we faced difficulty when trying to get this to work simultaneously the car movement; where the frog could only begin to move once the car reached the end of the loop. Therefore, we took inspiration from Gustavo Rodriguez’s frogger game and use a switch case statement (<https://uk.mathworks.com/matlabcentral/fileexchange/52883-frogger-game>) and used ‘If statements” to prevent the frog from moving out of bounds.

Car movement:

Initially we used an hgtransform function, but it was difficult obtaining data for the positions. As a result, we changed from using patch to rectangles for car. We made a loop to update the position of the cars. ‘If’ and ‘elseif’ statements were used to check when the rectangle blocks would go past a certain point. It was done in a ‘while’ loop so that the game could frequently check the position.

We later faced difficulty when trying to get the different rows of cars to move separately. At first, we had used a ‘for’ loop to move the cars, but this meant we couldn’t control the speed of an individual row of cars. Therefore, we used multiple if and else if statements and eventually used ‘randi()’ to make the cars have random speeds, within a given range. Also, a pause time was used to further control the speed and change the refresh rate of the game.

Collision:

Initially we came up with a method that included using poly shapes to detect overlapping of the cars and the frog by their matrix properties. However, it was not possible for us to detect if there was overlapping as we have created the cars using rectangles and could not identify a matrix as a property. Instead, we used Gustavo Rodriguez’s technique of using an equation to detect how far away the frog is from the cars, if there was a small distance less than that of what we specified, then the game would end.

Start-up screen:

We wanted to create a start screen to better organise our game, where instructions were given and where the player is able to choose the difficulty of the game. We created a figure in which contained these options, similar to the way we created the roads of the game. The way we integrated the options of difficulty levels was to use uicontrol which creates an interface. Using MathWorks documentation page for uicontrol, we found how to code a response for a pop-up menu selection. The issue we then faced was trying to retrieve the output once an option was selected to then be able to start the game accordingly to the difficulty level. We solved this by using a while and if loop, using the get code to retrieve the output and qualifying it to the outputs which were given if a certain option was chosen.

**Results obtained:**

Our recreation of the Frogger game showcases an arcade like experience for the user; with the vivid colours for the cars, the sound effects and varying levels of difficulty. Initially we had a very basic structure of the game with blocks, but later added a more engaging user interface. We decided to include different speeds of the cars through having difficulty levels, as we were able to manipulate the code to vary the speeds. Regarding the colours of the cars we were able to use a randi() function to generate random colours. The random function was also used with the music selection to increase the replay value of the game. Incorporating a start screen was vital as it allowed the user to read the instructions and select their desired level of difficulty. We faced challenges trying to link the start screen to the game itself, but we eventually solved this by using a ‘get’ command. Adding a win state was important for user satisfaction. Initially we faced issues with getting the frog to trigger the win state immediately after reaching the grass. This was tied to the ‘WASD’ key movements, so an extra key press was required to trigger the ‘if’ statement. This was then moved to a ‘while’ loop so that the ‘if’ statement would be checked on every refresh.

The game was then tested several times of different operating systems and errors were dealt with. Overall the additional features added to the game helped to produce a better overall product.

Music References:

Below is the music we have used in the game; the files can also be found in the Frogger.m file.

* Benny Hill theme:

https://www.youtube.com/watch?v=MK6TXMsvgQg

* Rammstein - Waidmanns Heil (instrumental with lyrics):

<https://www.youtube.com/watch?v=I_ArwyBt3PM>

* Undertale - 043 - Temmie Village:

<https://gamethemesongs.com/Undertale___043___Temmie_Village.html>

* Undertale OST: 005 – Ruins:

<https://www.youtube.com/watch?v=QyPR77rg1to>

* Undertale OST: 013 - Home (Music Box):

<https://www.youtube.com/watch?v=ANEsXjSiYxI>

* Undertale OST: 036 - Dummy!:

<https://www.youtube.com/watch?v=N3epEVMNJdY>

* Super Mario Bros (NES) Music - Underground Theme:

<https://archive.org/details/SuperMarioBrosNESMusicHurriedOverworld/Super+Mario+Bros+(NES)+Music+-+Underground+Theme.mp3>

* sound-frogger-hop:

<http://www.classicgaming.cc/classics/frogger/sounds>

* carstarthonkbackfire:

<https://www.freesoundeffects.com/free-track/carstarthonkbackfire-466330/>

* cheer2:

<https://www.freesoundeffects.com/free-track/cheer2-426825/>

* MOM GET THE CAMERA - MLG Sound Effects (HD):

<https://www.youtube.com/watch?v=8fsH1Rp_5bk>

* Gerudo Valley

<https://downloads.khinsider.com/game-soundtracks/album/legend-of-zelda-ocarina-of-time-original-sound-track/68%2520-%2520Gerudo%2520Valley.mp3>

For all YouTube links, we used https://www.onlinevideoconverter.com/mp3-converter to convert the videos to MP3

**How to play the game:**

- Download all files and folders. Please enable sound.

- Start the game by using Frogger.m and run the code to start.

- A menu screen will appear, please choose a difficulty which include:

1. Easy- Easiest mode for beginners.
2. Hard- A harder version of Easy mode.
3. Hardest- A mode to truly challenge the player.
4. Chaos- For the those who can accomplish the impossible (Epilepsy warning).

- Please do not close the game after selecting a difficulty, the menu start screen will close automatically and start the game.

- Using the WASD keys, guide the frog to the greener grass at the top of the screen avoiding the cars.

- The game will close upon successfully completing or failing the game.

**Mark Distribution:**

|  |  |
| --- | --- |
| **Members:** | **Marks:** |
| Alex Lai | 29 |
| Rhys Maher | 14 |
| Priyank Patel | 29 |
| Sonia Patel | 16 |
| Kevin Paul | 12 |