**Custom Project & Industry Link Competition Report**

Prepared for

**10.4HD M Task**

**&**

**Industry Link Competition**

By

**Hoang Minh Khue**

**Swinburne University of Technology**

November 27, 2022

**This report contains:**

* Introduction to the game Cyberpunk 7702
* Brief guide to make Front-end design (using Ruby, Gosu)
* Guide to make Back-end (AstraDb, Nodejs)
* Obstacles encountered and conclusion

1. **Introduction to the game Cyberpunk 7702**

* “Cyberpunk 7702” is a game designed to be my custom project to submit to my school, but it was improved to meet the requirements of Industry Link Competition. Therefore, this report and other items will be submitted to both.
* The name of the game was inspired by the famous game “Cyberpunk 2077” developed by CD Projekt. I think the color and design style were quite similar, so I decide to choose this name for my game.
* “Cyberpunk 7702” is a “Platformer 2D Shooter” game, which the player can move around, jump, and shoot using keyboard and mouse.
* The game contains 4 main sections (or stage):
  + Start Section – The first section that the player sees when opened the game. In this section, there is information about how to play the game and top scores; there are buttons to exit the game, reset data, enter game section, enter shop section, and see credits.
  + Shop Section – This section can be accessed by clicking on the “Shop” buttons in Start Section and End Section. This section allowed player to buy and change weapon, skin, and power.
  + Game Section – This section can accessed by clicking on the “Start” button in the Start Section or the “Restart” button in the End Section. It will take the weapon, skin, and power that were chosen in Shop Section. In this Section, the player can shoot at the enemies to earn coins and score.
  + End Section – This section can be accessed when the player died in the Game Section or clicking “Quit” button. This is where the score of the game and the most recent best score are showed and the buttons functioning the same as Start Section.

Link to demonstrating video:

Google drive: <https://drive.google.com/file/d/1DhArTgenH5oEFcGZ45U1iQbX6LCYZYSe/view?usp=sharing>

OneDrive: [demonstration\_video.mp4](https://liveswinburneeduau-my.sharepoint.com/:v:/g/personal/104221423_student_swin_edu_au/EbA1k59wYD5Bh9TiugyzF80BZv9FvGB2q3gIGe9pENPWqQ?e=4RAQVM)

1. **Brief Guide to make front-end (using Ruby, Gosu)**

* The code of the game is long and contains too much information to be provided in this report, so I will only explain some features that I found most challenging to make. Some may look simple in the code but required lots of researching and testing.
* Challenging features:
  + Rotatable gun for player and enemies
  + Physics (gravity, collision)
  + New Thread

1. Rotatable gun
   * + As a Platform shooter game, the game must have a player that can jump around with a gun that points to where the cursor is. Therefore, the required information to do this task is the x and y coordinates of the cursor and the point where the gun should rotate around (which could be the x and y coordinate of the player). Now, I have the required information; I need to turn these into a rotatable gun.
     + The required parameters for the gunText

       Description automatically generated
     + Math.atan2(@xpointat, @ypointat) return the angle needed in rad

Text

Description automatically generated

* + - rotate(@angle\_in\_deg, @x, @y) { draw functions to be rotate } this function rotates any draw function in the {} around @x and @y with @angle\_in\_deg degrees from the right horizon. Text

      Description automatically generated

1. Physics

The player can jump and move around so the game need some physic for it to feel right.

* + - Gravity: I only replicated gravity and not actually use the real gravity formular Text

      Description automatically generated

In this function, if the player’s bottom is not colliding with any ground, @gforce will gradually increase and player’s y coordinate increase by @gforce each frame.

* + - Collision: Every character in this game (player, enemies) and ground have attribute accessor to get its top, bottom, left, and right edges to detect collisionText

      Description automatically generated

Check if the player is colliding with a ground’s top edge:Text

Description automatically generated with medium confidence

1. New Thread
   * + When testing online functions, I realize that sometimes the Internet connection is not reliable; and if the game can only run if those online functions can run, then there is a high chance that the game would crash, so I decided to use new Thread for unstable functions
     + Implement: the following is the function to load top scores data from AstraDb to the gameText

       Description automatically generated

Any code in the {} will run simultaneously with the main code, and if there is any error, the main code can still run normally.

1. **Guide to make back-end (AstraDb, Nodejs)**

* To make the top score board of the game, I have to use many tools available. The tools that I used are:
  + Datastax – AstraDb: Storing, getting, and sorting data
  + Node.js – Express.js: Server-side and API control
  + Heroku: Hosting Node.js application

1. Datastax – AstraDb
   * + I sign in and follow the instructions on the <https://www.datastax.com/products/datastax-astra> website to create a database and a key space
2. Heroku
   * + Create a Heroku account
     + Follow instructions at <https://devcenter.heroku.com/articles/getting-started-with-nodejs#set-up> to install git and Heroku CLI
     + Type Heroku login to terminal to login to Heroku CLI
     + Type Heroku create to terminal to create a random name Heroku app
     + Type git clone “https://github.com/heroku/[APP\_NAME].git” to clone empty directory with [APP\_NAME]
     + Type cd [APP\_NAME]
3. Node.js – Express.js
   * + Setup:
       - Install Node.js at

<https://nodejs.org/en/download/>

* + - * Type npm install -s body-parser cassandra-driver cors express into terminal to install the needed library for Node.js
      * Type npm init to initialize Node.js project
      * Create index.js file
      * Create a folder name secure, then copy the secure bundle downloaded from AstraDb website to it
    - Create the Node.js app
      * In the index.js file, follow instruction at <https://docs.datastax.com/en/astra-serverless/docs/connect/drivers/connect-nodejs.html#_connecting_with_node_js_legacy_driver> to connect to AstraDb
      * To create a Table in the key space, I write the following code to make an API to create a table using NoSQL and only call the API once

app.get("/createtable", async function dosth(req,res) {

        const result = await client.execute("CREATE TABLE users\_info.best\_scores(bestscore INT, type INT, username VARCHAR, PRIMARY KEY((type),bestscore)) WITH CLUSTERING ORDER BY (bestscore DESC);")

        res.send(result)

    })

This create a table with 3 columns bestscore, type, and username with order by bestscore descending value

* + - * To insert data, I made the following code and insert data into the table by making a POST request

Code from server to receive data

app.post("/insertdata",bodyparser.json(), async function dosth(req,res) {

        let username = req.body.playername

        let bestscore = req.body.bestscore

        const result = await client.execute(`INSERT INTO users\_info.best\_scores(username, type, bestscore) VALUES (?, 1, ?);`,[username, bestscore],{prepare:true})

        res.send(result)

    })

Code from client to send data

def savedata(name, bestscore)

        Thread.new {

            posturl = HOST + '/insertdata'

            res = HTTParty.post(posturl, :body => {

                :playername => name,

                :bestscore => bestscore,

            }.to\_json,:headers => { 'Content-Type' => 'application/json' })

        }

    End

* + - * To get data, I made the following code and get data by making a GET request

Code from server to receive request

app.get("/getdata", async function dosth(req,res) {

        const result = await client.execute("SELECT \* FROM users\_info.best\_scores WHERE type = 1 ORDER BY bestscore DESC")

        res.send(result.rows)

    })

Code from client to request for data

def loadrank

        Thread.new {

            geturl = HOST + '/getdata'

            res = HTTParty.get(geturl)

            res = JSON.parse(res.body)

            @num\_of\_records = res.length

            if(@num\_of\_records > 5)

                count = 5

            else

                count = @num\_of\_records

            end

            rank = Array.new

            for i in 0..count-1

                rank << Record.new(res[i]["username"], res[i]["bestscore"])

            end

            @rank = rank

        }

    end

1. **Obstacles encountered, Feedback, and Conclusion**

* As a beginner, I have little to no knowledge about and experience with using tools such as AstraDb, Heroku, Ruby language, and networking; I was struggling learning the syntax of a new language or library and it took me quite an amount of time.
* I would love it if Datastax could provide more example on NoSQL command such as CREATE TABLE users\_info.best\_scores(bestscore INT, type INT, username VARCHAR, PRIMARY KEY((type),bestscore)) WITH CLUSTERING ORDER BY (bestscore DESC); because there could be confusion between SQL and NoSQL syntax. I think Datastax should give more real use case of NoSQL answering question like “How to create a table with name and descending score?” and “How to insert data into table?”. They are simple question, but I think they are beneficial for newbies.
* In conclusion, I made the game “Cyberpunk 7702” for the community to play and learn from it; it also help me practice programming and research skills. AstraDb from Datastax really helped me in making the game since the “Top Scores” board can increase the competitiveness among players.

**-----END-----**

**By**

**Hoang Minh Khue**

**Swinburne University of Technology**