Project Documentation

Author: Oratile Diale

[27 September 2023]

Project Title: Astro-Slither Game

Project Team:

-Keokeditse Ofentse Ndala

-Siphosethu Malinga

-Ralph Ralefatso

-Oratile Diale

Member’s roles:

Keokeditse Ofentse Ndala – The group leaders, would organise our meeting times. The group leader also had to oversee the roles of every member in the group. He also partook in the coding of the game.

Siphusethu Malinga – produced part of the coding (score-keeping)

Rothang Ralph Ralefatso – Coding of the game; created the graphic of the game, controls,..

Oratile Diale – All the documentation of the game (Credits, proposal, ReadMe, documentation), multiple snakes moving randomly( the code unfortunately didn’t fuction accordingly)

**Background and Motivation**

The original snake game originated in the 1970s and was first created by Grelim Industries(programmer) who developed the game for early arcade machines. The original snake game which is the one Grelim created was called the “Blockade” version. The game found its mass audience with the Nokia cellphone where players would leave a trail behind them whenever they turned using the arrow keys to move the snake.

The version that we as a group was interested in implementing is the Slither.io version. This is the version that allows multiple video game. The original creator of this version is Steve Howse. He started this game in 2016 when he was facing some financial problems. Initially the only development at the time was in the Adobe Flash however he did not want to use this method and therefore put a hold on these ideas for some time.

When he found out that WebSocket, which is a low-latency protocol supported by dominant browsers, was stable enough to handle numerous players at a time and run an HTML game. It took Howse 6 months to develop the Slither.io game and was released on March 25, 2016 to browsers and iOS. (Slither.io, 2023) . By July 2016, the browser version website was ranked Alexa as the 250th most visited site worldwide, but the experienced a decline in popularity, with the ranking below 1000 by October 2016 before it mostly constant at approximately 1700 by January 2017. (Slither.io, 2023)

We as a group have decided to create it for entertainment purposes and the reason for specifically choosing a snake game is because not only is it inclusive(as snake games are usually played by anyone from the ages as young as 2, to teenagers and even beyond that!) but it is also favoured but many people. What further makes this game inclusive is that it does not promote any violence, or anything of that sort.

**Problem Statement**

This aim of this is this game is to keep the snake alive(moving) for as long as possible. The game has a low complexity

**Approach**

The game has a relatively simple objective, in that the aim is for the user to grow the snake as long as long as possible and thereby also increasing the player’s score. This may be achieved by the snake eating “food” which is randomly placed in certain random tiles. The snake is considered to eat the food when its head collides with the food. If the food is eaten, it is then becomes randomly placed in another (or the same) tile. If the snake happens to collide into itself(which often happens when the snake has grown extremely long) or even a wall, then the game is over as the player dies.

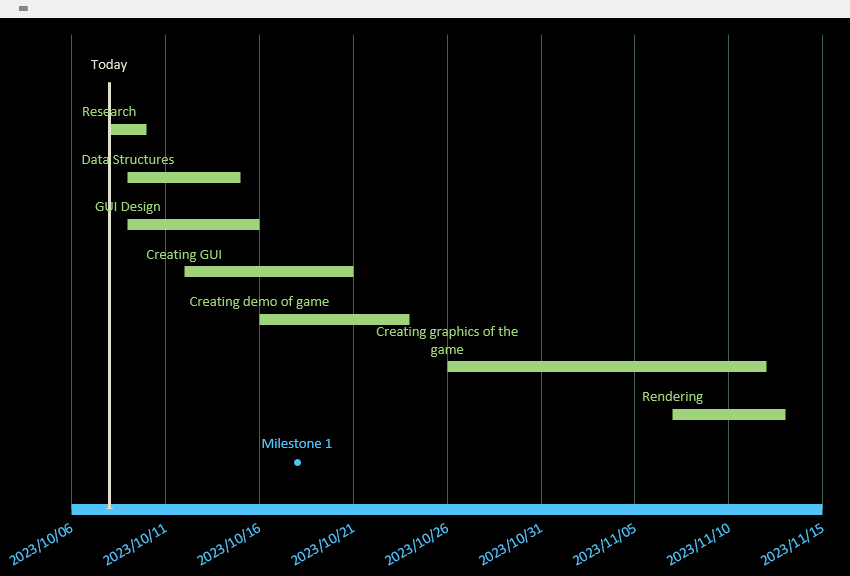
When the snake grows extremely long, this makes the game more thrilling and harder as it makes clear paths for the snake to turn to quite limited and also making it more difficult for the snake to move easily/with freedom and also increases the chances of the snake having collisions- especially into itself. The player is able to control the snake’s movement in 4 directions(upwards, downwards, west as well as east). Inasmuch as the game’s objective is simple, it is still able to keep one on their toes as they hope and anticipate to keep the snake alive for as long as possible.

**SDLC**

We as the group would meet at least once a week and discuss what we were able to accomplish from the last time we have met and if there was something one was struggling with, then shed some light with each other as to how that might be solved. Additionally, we would also sometimes do some coding during these sessions as well as to assign/discuss way forward from where we would currently be.

We have decided to use the Gantt chart in order to track as well as keep record of our work. This model would guide us in terms of keeping time and our tasks. We would follow the model to fulfil the tasks which need to be fulfilled keeping with the time frames we have set at the beginning. Our meeting times were also set in order to accommodate and follow the model.

We chose to use this chart as it is easy to use, understand as well as to manage controls(which includes planning and tracking). The model also worked well as this was not a huge project. The technology involved includes Java and a portable electronic gadget(i.e. laptop,..). Below is the chart for our project, which shows the tasks that need to be started/completed in a certain time frame.



Below are the classes we used to make the game functional. These classes include the Snake and its parts, the food which the snake can eat, the grass, the gameview(the display of the game); how the game looks.

|  |
| --- |
| Gameview |
| -boardWidth: int  -h: int  -w: int  + sizeofmap: int  -bmgrass1 :: bitmap  -bmgrass2: bitmap  -bmsnake: bitmap  -bmfood: bitmap  - snake:Snake  - mx: float  - my: float  + play1: mediaplayer  -gameover:Boolean  -arrgrass: ArrayList  - handler: Handler  -apple: food  -scoretextview : TextView |
| +gameview(Context context, @nullable Attributeset attrs)  +onTouchEvent(Motionevent event): Boolean  +draw(Canvas canvas):void  +randomapple(): int[]  +hitwall():void  +end(Boolean gameover):void |

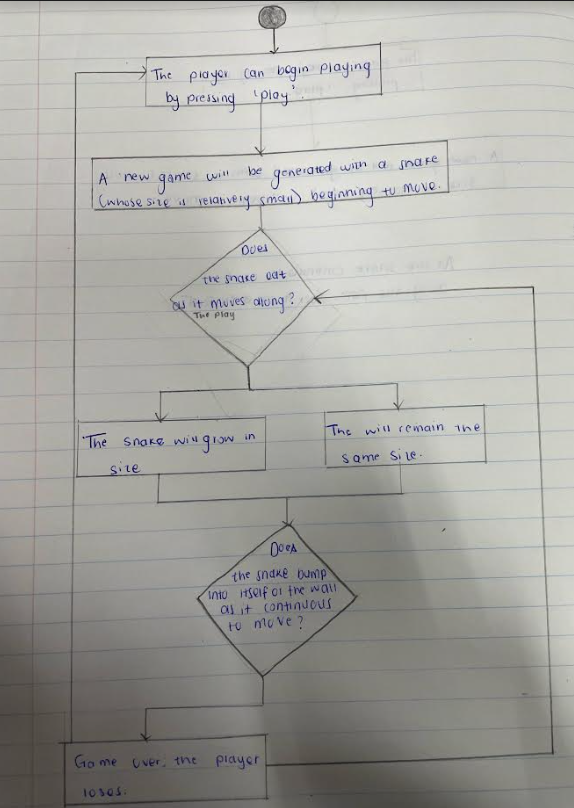
|  |
| --- |
| Snake |
| -x:int  -y:int  -length:int  -arrsnakeparts:boolean  -move\_right :boolean  -move\_left : Boolean  -bm ,bm\_head\_up,bm\_head\_down,bm\_head\_left,bm\_head\_right,bm\_body\_vert,bm\_body\_horiz, bm\_body\_top\_right,bm\_body\_top\_left,bm\_body\_bottom\_right,bm\_body\_bottom\_left,bm\_tail\_up,bm\_tail\_down,bm\_tail\_left,bm\_tail\_right : bitmap  - move\_top:boolean  -move\_bottom:boolean |
| + Snake(Bitmap bm , int x , int y, int length)  +update(): void  +draw():void |

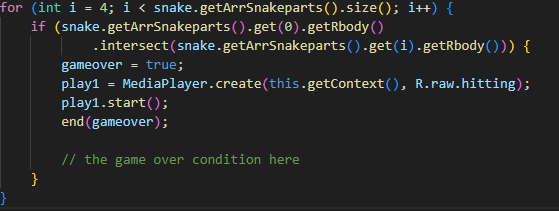
|  |
| --- |
| Food |
| -bm: bitmap  -x: int  -y: int  -r: rect |
| +food(bitmap bm, int x, int y)  +getBm():bitmap  +setBm(Bitmap bm):void  +getX(): int  +setX(int x)  +getY(): int  +setY(int Y): void  +getR(): Rect  +setR(Rect r): void  +draw(Canvas canvas): void  +reset(int nx, int ny): void  placeFood()  actionPerformed |

|  |
| --- |
| Grass |
| -bm: Bitmap  -x:int  -y:int  -width: int  -height: int |
| +grass(Bitmap bm, int x, int y, int width, int height)  +getBm(): Bitmap  +setBm(Bitmap bm): void  +getX(): int  +setX(int x): void  +getY(): int  +setY(int y):void  +getwidth(): int  +setWidth(int width): void  +getHeight(int width): void  +setHeight(int height): void |

|  |
| --- |
| snakeParts |
| -bm:Bitmap  -x: int  -y: int  -rbody: Rect  -rtop: Rect  -rleft: Rect  rright: Rect |
| +Snakeparts(Bitmap bm, int x, int y)  +getBm(): Bitmap  +setBm(Bitmap bm): void  +getX(): int  +setX(int x): void  +getY(): int  +setY(int y): void  +getRbody():Rect  +setRbody(Rect rbody): void  +getRtop():Rect  +setRtop() : Rect  +getBottom() : Rect  +setRbottom(Rec rbottom): void  +getRleft(): Rect  +setRleft(): void  +getRight():Rect  +setRright(): void |

**How the game works:**





The provided code snippet above checks if the snake has collided with itself or not. If the snake collides into itself, the game would then be over.

Below is a snippet of the move() method which controls the movement of the snake and define snake behavioiur.



Below is one of the codes that gave us a bit if a hustle to get working but we however was able to get it to work at the end. The code converts the keyboard controls to touch.



Lastly, another one of the codes that are quite interesting includes the ‘placeFood()’ method-provided below- which places food randomly on tiles which the player may eat. If the snake happens to eat this food, the snake will grow larger/longer.

public void placeFood(){

food.x = random.nextInt(boardWidth/tileSize);

food.y = random.nextInt(boardHeight/tileSize);

}

As stated in our motivation for creating the game, initially; when we thought of creating the snake game, what we had in mind was not the classic snake game but rather instead of having one snake, it be filled with multiple snakes, this however seemed to have become a problem as no one was able to get all the different snakes to move in different directions without causing a havoc. Have we been able to accomplish this, this was going to make the game even more interesting and complex. I believe however, that have we had more time, we would be able to accomplish this. Another one of the challenges we faced was making fixed controls which is converting the keyboard controls to touch but we however was able to solve that.

Some of the milestones we have faced is mainly that there was so much to learn in order to be able to create the game this includes, i.e. creating graphics,… . This also had to be done in constricted time.

**References**

Angelos, A. February 2021. This was the dawn of a new use for the mobile phone, and a game that would instantly turn into a phenomenon.

Patkar, M. (2017). A beginner’s guide to slither.io and the bets tips and tricks. Retrieved from

<https://www.makeuseof.com/tag/slithr-beginners-guide/>

Slither.io (2023, October 07). Retrieved from https://en.wikipedia.org/wiki/Slither.io

Switzer, E. (2019, July 09). Where Slither.io came from and why is so popular. Retrieved from The gamer: https://www.tehgamer.com/Slitherio-history-success-guide-stratgy/