

Computer Games AI Assignment 2

UWS UNIVERSITY OF THE
WEST *of* SCOTLAND

Summary

1. Introduction

2. Work Done

3. CV analyses

4. Bonus

5. References

INTRODUCTION

We were given a task to recreate a game with the code we have been given.

OBJECTIVE POINTS:

Those objectives ensure points if done properly:

- Creating a score, lives, and a high score.
- Resetting the game after it is over.
- Creating bonus zones.
- Turning those zones to traps upon contact.
- Reduce lives (player) and augment the score (enemy) in case of contact with trap.
- Creating a sound for the different interactions of the game.
- Prey Respawn Away from the collision.
- Analyses of OpenCV modifications.
- This report.

WORK ACHIEVED

I succeeded in creating everything that was asked except the thread.

Here is some main explanation on the most important parts for me:

```
void Respawn(int w, int h)
{
    float rand1x = (float)(rand() % (h + ((int)this->position_.x + 51))); //from the collision point radius 50 to height
    float rand2x = (float)(rand() % (((int)this->position_.x - 50) + 1)); //from 0 to the collision point radius 50

    float rand1z = (float)(rand() % (w + ((int)this->position_.z + 51))); //from the collision point radius 50 to width
    float rand2z = (float)(rand() % (((int)this->position_.z - 50) + 1)); //from 0 to the collision point radius 50

    int randomx = rand() % 2;
    int randomz = rand() % 2;
    this->position_.x = randomx == 0 ? rand1x : rand2x;
    this->position_.z = randomz == 0 ? rand1z : rand2z;
}
```

Respawn is a kinematic function that use 2 Randoms (from rand() function which is far from being the best random function but... eh it works) 1 random controls the whether the respawn will be after (spatially speaking) or before the collision. The 2nd random (divided in 4) is to take a number between the beginning of the map to the collision point minus a 50 gap space radius (AWAY we said !) or from the collision point to the end of the map plus a 50 gap space radius.

```
class Ground
{
public:
    Ground(const float z, const float x, int size = 20, int size_less = 5)
        : k_{ {x,0,z},{float}0,{0,0,0},0 }, col_{ GREEN }, size_{ size }, size_l{ size_less }, touched{ 0 }, inside_bonus{ 0 }, inside{ 0 }

    Kinematic k;
    raylib::Color col;
    int size;
    int size_l;
    int touched;
    int inside_bonus;
    int inside;

    void draw(int screenwidth, int screenheight)
    {
        ai::Vector2 pos{ k_.position_.z, k_.position_.x };
        if (touched == 1)
        {
            ai::DrawCircleLinesV(pos, size - size_l, col);
        }
        else
        {
            ai::DrawCircleLinesV(pos, size, col);
        }
    }
}
```

Ground is the class for the bonuses/ black holes.

It checks a lot of things, if we are in the bonus zone or in the black holes zone, if we touched the ground in question. It also saves the place of the ground, its colour and the size we want to attribute it (possible to change it in the main function at the creation of the ground).

```
ction for the collision with the player, 1st hit grants bonus then the ground goes black, beware !
d collision(Kinematic& ship_k, raylib::Sound& sound1, raylib::Sound& sound2, int w, int h, int& score, int& life, const float collision_d)

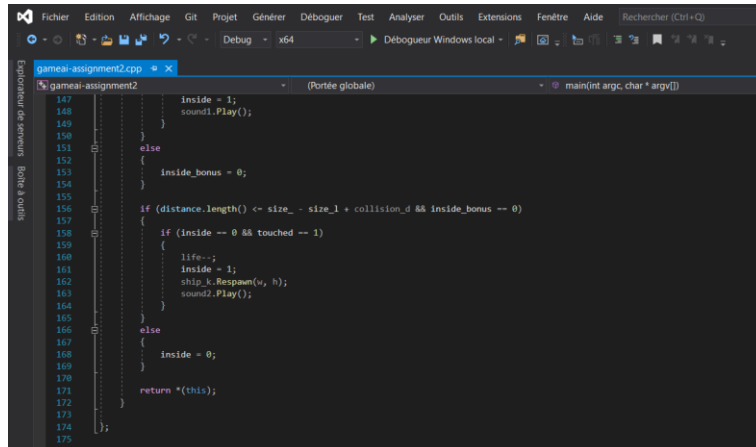
auto distance = this->k.position_ - ship_k.position_;
f (distance.length() <= size_ + collision_d)

    if (touched == 0 && inside == 0) // no == operand for col_, no strength to make one (not even sure we can take the RGB)
    {
        score += 1000;
        this->col_ = BLACK;
        touched = 1;
        inside_bonus = 1;
        inside = 1;
        sound1.Play();
    }
else
    inside_bonus = 0;

f (distance.length() <= size_ - size_l + collision_d && inside_bonus == 0)

    if (inside == 0 && touched == 1)
    {
        life--;
        inside = 1;
        ship_k.Respawn(w, h);
        sound2.Play();
    }
}
```

Here you can see how the collision works, I look at the distance between the ship in question and the ground. If it is in, I check if it was not already inside. If it wasn't then we enter it. Meaning the player can't lose more lives or gain more point by staying longer in it.



```
gameai-assignment2.cpp
147         inside = 1;
148         sound1.Play();
149     }
150     else
151     {
152         inside_bonus = 0;
153     }
154 }
155
156 if (distance.length() <= size_ - size_l + collision_d && inside_bonus == 0)
157 {
158     if (inside == 0 && touched == 1)
159     {
160         life--;
161         inside = 1;
162         ship_k.Respawn(w, h);
163         sound2.Play();
164     }
165     else
166     {
167         inside = 0;
168     }
169 }
170 return *(this);
171 }
172
173
174
175
176
```

I check if we're still inside at the creation of the black hole, then the player don't die because the black hole isn't fully created yet (you don't want the player to die right after he took a bonus right ?!) then if the player leaves the bonus, the black hole is fully created beware !

If you touch the black hole, you die and respawn somewhere on the map.

The enemy collision is simpler. If he touches the black hole he dies and respawn.

The rest of the code is easy to understand but feel free to check it.

OPENCV ANALYSIS

CV::RESIZE

In the Resize function fx and fy make the OpenCV camera image scale, bigger values get bigger % of image, 1 for example would give the original size of your camera image.

At beginning we start with 0.5,0.5. In comparison, 0.666×2 ($0.666 fx$, $0.666 fy$) turns faster. It would be nice to avoid a stiff neck! At 0.8 and higher, it gets more difficult to control our character... Moreover, the game seems "slower, the movements are jerky.

In the other hand, 0.333×2 turns out to be much more "us" meaning the character will imitate your movements. You will need to turn your shoulder to make a bit more than a quarter turn. At 0.2 and lower, it gets harder to control the character, the character response to your action are too slow and you always need to be moving to survive.

These results, however, look logical when we compare to the meaning of the Scales fx and fy . Indeed, if the image is bigger, we receive more information and the smallest movement is visible, hence the augmentation of the movements' speed. In the opposite way, if we have less information (smaller image) we only detect the important moves and it is more precise but slower.

So overall, yes, we can change the value from 0.5 to something else but control will be changed (I personally find the 0.333×2 "correct" on terms of control and performances).

CV::MEAN

Everything is possible with a computer isn't this the beauty of it? To be more specific: No, you can't use the premade thresholds of OpenCV in the mean function. However, threshold can be done by different ways (erode, dilate, etc). Maybe a mask for thresholding could be made and then surely be applied on the mean function. If we are limited to cv premade possibilities, then no this isn't possible.

CalcOpticalFlowFarneback

The parameter `pyr_scale` creates pyramids of layers for the analysed image, over 0.5 it slows everything down, the control seems less good. Under 0.5 in the other hand the game isn't slowed, it seems that the fluidity of movements is better, but it might be just an impression.

Levels don't make much change except in a slight delay in answer the more we add to the number (which is explained by the fact that we have more extra layers created so more layers to go through I guess ?)

Winsize helps for movement detection. Add 1 more zero and moving your head too far will make your character go 360. 20 (so 5 more than the basic) seems to help get a better control over the character without going down on FPS.

For the Iteration, the more there are the more the control is easy but a problem is that the time of “resetting middle face” will also be faster, meaning if you turn right for more than 3sec the program will consider your right profile face to be you looking at the camera ! (this is at 10levels).

Poly_n seems to affect the control in a way I can’t understand. It seems to slow down the reaction when higher? Maybe because its purpose is to smoothen the image?

Poly_sigma seems to affect the “wave” coming back to its initial stat faster the smaller poly_sigma is. For example, if I go full right, at 1.5 the character will go back left a bit when it goes straight. At 1.1, he will go back left less. But the effect seems really slight.

Flag property doesn’t seem like a parameter we should change.

FRAME RATE:

The basic frame rate is: 7 FPS at the creation and then 29-31 FPS for the rest of the game after the creation. (this is also influenced by the light in the room, in the “dark” general FPS drops at 15)

On resize change: at 0.5 and under there are no changes but higher (2/3, 0.8) the FPS decreases to 23~ FPS at 0.666 and 21~ FPS (after creation) at 0.8 (which explains the jerky movements of the game).

Pyr_scale in CalcOptical influences FPS, at over 0.5 the FPS drop at 25fps.

Levels in CalcOptical doesn’t seem to affect the game frames much, we can see a difference of only 1 or 2 FPS compared to the basic value we’ve been set with.

Winsize in CalcOptical will drop your FPS to 24~ when you add a 0.

No change on FPS from Iterations, Poly_n or Poly_sigma in CalcOptical.

Threads

I started to create one, but I didn’t manage to change the values in the main by reference, I then tried to pass by a structure, but it didn’t work either.

The 1st time I tried it didn’t work but the FPS went up to 60FPS (thread detached) and throw an abort. The structure method didn’t work at all (guess std::thread doesn’t work as pthread !)

BONUS

As bonus, we added a few options:

There is a main theme music playing in the game.

The duration of the game is shown.

Lines have been added under the texts to give a better visual.

I wanted to add different other bonuses linked to the ground (different buffs depending on the bonus) but the thread took me too long and ... I did not even finish it...

For more information on the code or work achieved, the code itself is commented.

The exe is in the zip too.

REFERENCES

Those are all the sources we used to understand and recreate the game:

- Course Slides
- <https://github.com/RobLoach/raylib-cpp>
- <https://en.cppreference.com/>