OBJECT ORIENTED PROGRAMMING

PROJECT REPORT

Coder : Nguyễn Phương Bảo & Thái Phát Tài

Coordinator : Nguyễn Đức Bình

I. Task distribution

1. Nguyễn Phương Bảo : Map and HUD.
2. Thái Phát Tài: Player, Enemy and Bullet
3. Nguyễn Đức Bình: Drawing and write the report.

II. Introduction

In this project, we are so far ago desired to make a game, and we had so many idea about the character and the enemies as well as the gameplay we want to make. First we thought coding a game would not be so hard, but then a lot of new knowledge about the extern software SFML had proved that we're wrong. But anyway after simplify our ideas, we decide to make this one.
It can be called as a dungeon game, and here some general point to other games.In conclusions, in this project.

General points which is in common with others game:

- Final aim is to beat the Boss at each stage.
- Enemies will spawn random.
- Execute the code with extern software **(SFML).**
- When the player beat the second boss the game is finish.
- Player is manipulated by the user to move and fight.
- Player can be heal and mana-filled by the collide with the pickup.

Additional features: drawings and our player can dash

III. SFML functions and variables

In this project, we did not use much of hard understanding algorithm,vice versa we used SFML(Simple and Fast Multimedia Library) which provides us a simple interface, to ease the development of our game. Some functions and variables can be listed as :

-To create and display the Player and Enemies:

- Texture : a variable that hold the texture of the picture
- Sprite : a variable that hold the picture
- texture.loadFromFile() : to store the texture to the variable name texture
- sprite.setTexture() : to set the picture's texture
- sprite.setOrigin() : to set the origin of the sprite
- sprite.setPosition() : to set the position of the picture in terms of the window
- sprite.getPosition() : to return the current position of the sprite
- sprite.setRotation() : to rotate the sprite to a specific direction
- sprite.move() : to move the sprite
- sprite.getGlobalBounds() : a functions return global bounding rectangle of the entity
- sprite.getGlobalBounds().intersects() : a function to check collider
- sprite.setTextureRect() : read the texture in terms of the value we have set can also be understand as

- To know what happen to the window at the time we use the variable Event, in our project there are four type of event Keyboard, Mouse, Closed and Resized
    - isKeyPressed() : to check if the key is pressed
    - MouseButtonPressed() : to check if the mouse is clicked
    - Closed and Resized : whether the window is closed or resized.

- To create and display thing on a window:
    - RenderWindow : a default constructor to create a window
    - window.isOpen() : which will return a boolean value
    - window.clear() : to clear the entire target with a single color(RGB).
    - window.display() : to display what has been rendered to the window so far.
    - window.draw() : to draw a drawable object to the render target
    - window.close() : to close the window
    - window.mapPixelToCoords: in general this function help us to find the mouse position
    - window.pollEvent() : to check if there is any event happen to the window

IV. Implementation

1.Animation

First when we want to make characters, there are player, enemy and boss. At the very first thought, our main character would be really easy to manipulate, just moving around, dash and cast skill, the enemy will chase the main character and the boss just moving randomly. So to make them move smoothly, here we have the Animation class which make them look better.

Animation is to perform a movement so in another way, it can be called as to change from this frame to another frame till the last one and then begin again.
Because i want to use a picture contain a movement with multiple drawings of the character so in this Animation class's protected session:

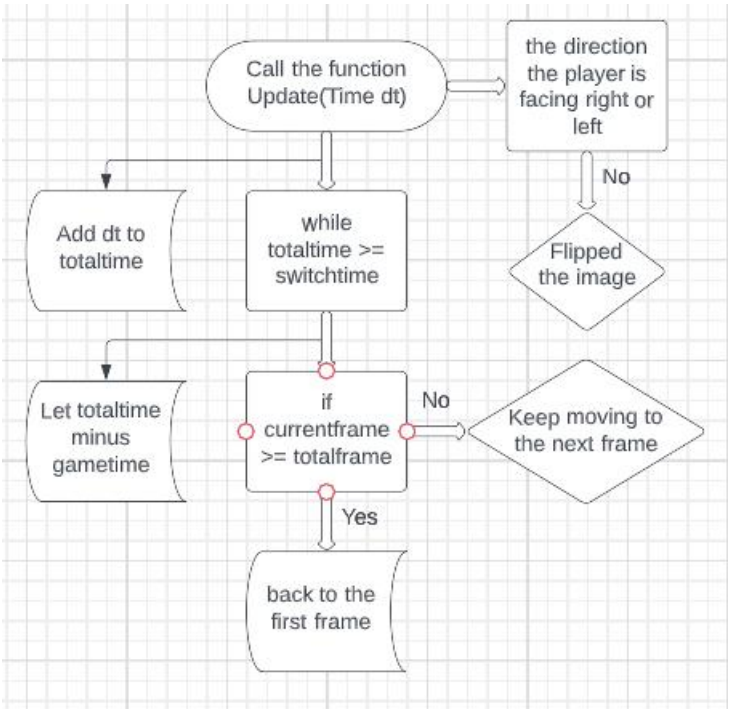
```
class Animation {
protected:
    short totalframe; //is the number of drawing in a picture
    short currentframe; // is the current drawing in the movement
    short framewidth, frameheight; // the size of the frame

    //totaltime can be as the gametime which is how long has the game run
    //switchtime is for changing the frame
    float totaltime,switchtime;

    //Direction of the movement
    bool faceright;

    Texture texture;
    Sprite sprite;

public:

    Animation();
    Animation(const short framewidth, string& texturelocation, float switchtime);
    // Make the effect
    //deltatime is the time between two images
    void Update(float deltatime);
};
```

And in the public session there are a default constructor just to set those Animation's protected variables to default, and a constructor has parameters:

```
Animation::Animation(const short framewidth, string& texturelocation, float switchtime){
    this->switchtime = switchtime;

    totaltime = 0.0;
    currentframe = 0;

    texture = texture;
    totalframe = texture.getSize().x / framewidth;

    this->framewidth = framewidth;
    this->frameheight = texture.getSize().y;

}
```

And final the major part in Animation, the Update function:

```
void Animation::Update(float gametime) {
    totaltime += gametime;//assign gametime to totaltime

    //Changing the frame
    if (totalframe != 1) {
        while (totaltime >= switchtime) {
            totaltime -= switchtime;
            //Make it back to the first frame after finish a movement
            if (currentframe >= totalframe) {
                currentframe = 0;
            }
            else
                currentframe = (1 + currentframe) % totalframe;// move to the next frame but just during totalframe
        }
        totaltime++;
    }

    if (faceright)
    {
        sprite.setTextureRect(sf::IntRect(currentframe * framewidth, 0, framewidth, texture.getSize().y));
    }
    else
    {
        //Read the texture from right to left using negative numbers.
        sprite.setTextureRect(sf::IntRect(framewidth * (1 + currentframe), 0, -framewidth, texture.getSize().y));
    }
}
```



2. Player

So then here we have our player class:

```
class Player : public Animation
{
protected :
    short Health, Speed, Damage; //status

    bool dead; // the player is dead or not
    bool direction; // true is up vice versa
    Time lasthit;// the last time the character was hit

    int type_of_movement;//1 is walk, 2 is attack

public:

    //Default constructor
    Player();
    //Reset the status of the player to default
    void Reset();
    //Manipulate the player
    void Moving(float deltatime,RenderWindow& window);
    //draw the player to the window
    void draw(RenderWindow& window);
    //get the direction left or right, up or down
    bool get_faceright();
    bool get_direction();
    //get player position and size
    Vector2f getpos();
    Vector2f getsize();
    //get the sprite global pounds
    FloatRect getRectPos();
    //if the player is hit
    void hit(Time hittime,short damage);
    //get the last time the player was hit
    Time getlasthittime();
    // for the status
    short get_health() const;
    short get_damage() const;
    short get_mana() const;
    //check if the player is dead or alive
    bool isdead();
    //set the status
    void set_health(unsigned short damage);
    void set_damage(unsigned short damage);
    void set_speed(unsigned short speed);
    //if the player collide with the pickups
    void pick_up(Pickup pickup, int type);
    //can be understand as set pos for the player
    void move(Vector2f pos);
```

There are 3 kind of status :

-Health : represent for how many hits can the player catch

-Speed : represent for how fast does the player move

-Damage : represent for the player's ability to deal damage

And in public session here we have:

- A default constructor
- 2 bool functions to get the player's direction and 1 to check if the player is alive or dead
- 3 functions to get player's status, 3 functions to set them and 1 to reset them
- A function to check the collision with the pickup
- A function to set player's position
- 2 function to get players's size and position

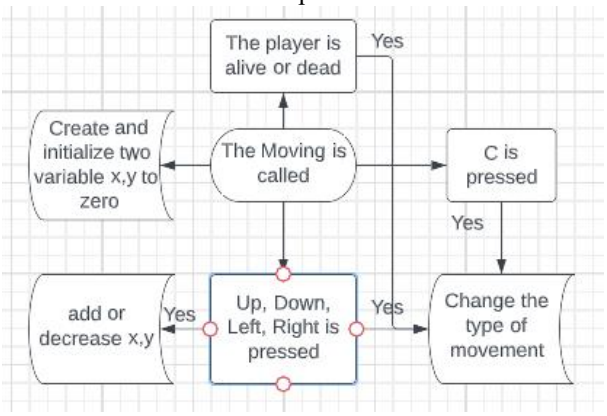
And the major one is to manipulate the player - Moving:

```
void Player::Moving(float deltatime,RenderWindow& window)
{
    Vector2f pos;
    pos.x = 0.0;
    pos.y = 0.0;

    //walk
    if (Keyboard::isKeyPressed(Keyboard::Left)) {
            pos.x -= Speed * deltatime;
            type_of_movement = 1;
            faceright = false;
    }

    if (Keyboard::isKeyPressed(Keyboard::Right)) {
            pos.x += Speed * deltatime;
            type_of_movement = 1;
            faceright = true;
    }

    //walk up and down
    if (Keyboard::isKeyPressed(Keyboard::Up)) {
        pos.y -= Speed * deltatime;
        type_of_movement = 1;
        direction = true;

    }
    if (Keyboard::isKeyPressed(Keyboard::Down)) {
        pos.y += Speed * deltatime;
        type_of_movement = 1;
        direction = true;
    }
```

First we create a variable Vector2f to store how the distance we want to move .
Then we will manipulate the player with four key Left, Right, Up, Down.
With key C the player will attack.

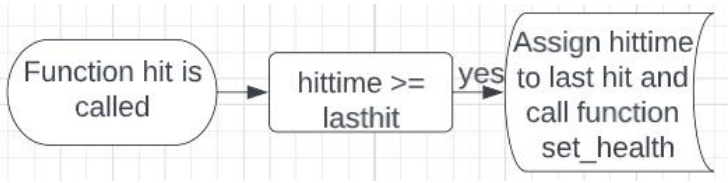
```
//attack
if (Keyboard::isKeyPressed(Keyboard::C)) {
    type_of_movement = 2;
}

sprite.move(pos);
Update(deltatime);

if (pos.x == 0 && pos.y == 0 ) {
    type_of_movement = 1;
}

//dead
if (dead == true) {
    type_of_movement = 0;
}
```

Here is the pseudo code



And on the other hand, we don't want our player die instantly when it collide with the enemies so we have the function hit:

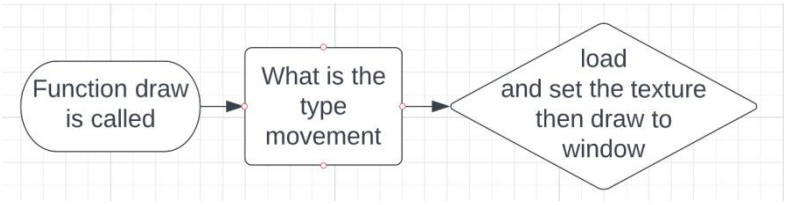
```
void Player::hit(Time hittime,short damage)
{
    if (hittime.asMilliseconds() - lasthit.asMilliseconds() > 1000) {
        lasthit = hittime;
        set_health(damage);
    }
}
```



Variable hittime can be also called as the moment when the enemies first touch the player and when it happen lasthit which is 0 will be assigned with hittime. Then the second time the player get hit only count after that moment 1s.

And to display our player, we need draw function:

```
void Player::draw(RenderWindow& window)
{
    switch (type_of_movement)
    {
    case 0:
        texture.loadFromFile("char/maindead.png");
        break;
    case 1:
        texture.loadFromFile("char/main.png");
        break;
    case 2:
        texture.loadFromFile("char/mainattack.png");
        break;
    }

    sprite.setTexture(texture);

    window.draw(sprite);
}
```



As we've already known, there are 3 kind of image of the player: walk, attack and dead. So here we use switch function to load the texture, then we set the sprite texture then we draw it to the window.

3. Enemy

Enemy is player alike, it has Speed, Health and Damage but we don't manipulate it, enemies will try to chase the player.

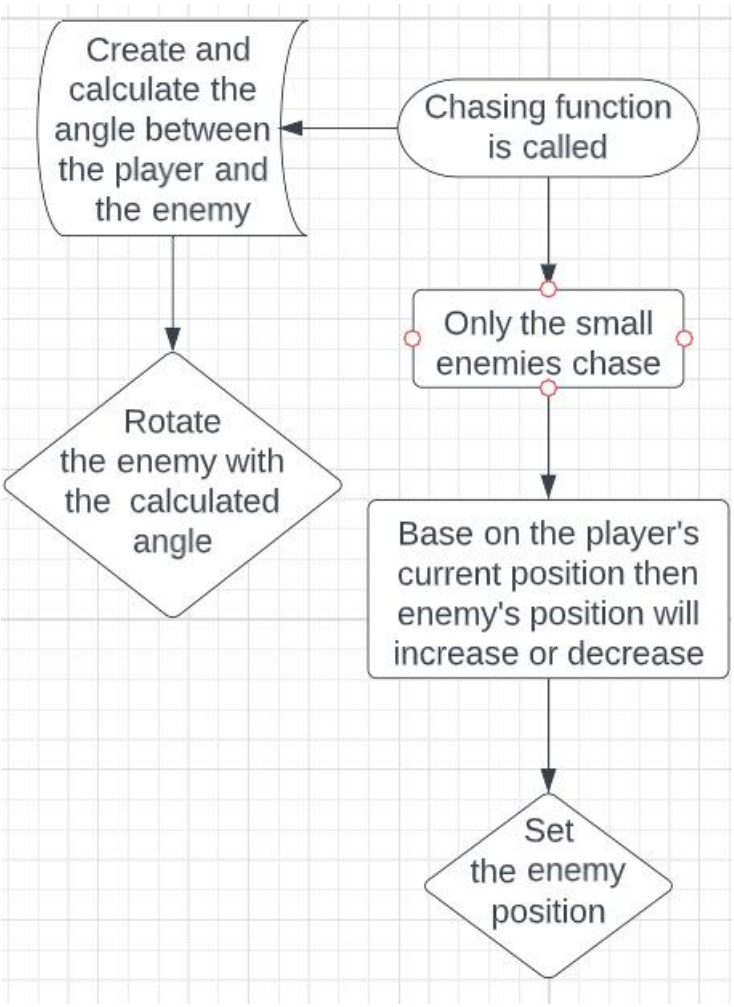
```
class Enemy : public Animation
{
private:

    short type;
    Vector2f pos;//position of the enemies
    Vector2f frame;//framesize - width and height
    short status[3]; // 1 health, 2 speed, 3 damage
    bool isDead; //if the enemy is dead or alive

public:
    //Default constructor
    Enemy();
    // Update/Chase the player
    void Chasing(float dt, Vector2f playerpos);
    // when the enemies is hit
    void getHit(short damage);
    // to check if the enemies is dead
    bool dead();
    // draw the enemies
    void draw(RenderWindow& window);
    //set position for the enemies
    void move(float x, float y);
    //Spawn enemies
    void spawn(Vector2f pos, short level);
    //Get enemy's position
    Vector2f getpos();
    //Get enemy globalbounds for checking collision
    FloatRect getRectPos();
    //Get enemies framesize
    Vector2u getsize();
    //Get enemy damamge
    short get_damage();

};
```

Here we have:

- Of course a default constructor
- A bool function to check if the enemy is alive or dead : dead()
- 3 functions to get enemy's position, globalbound and size. : getpos(), getRectPos(), getsize()
- A function to set enemy's position : move()
- A function to get enemy's damage : get_damage()
- A function to reduce the health when collide with bullet : gethit()
- A function to draw the enemy to the window : draw()

And as we have discussed, the enemy will chase the player so we will need a function have player's position as parameter, here we have Chasing

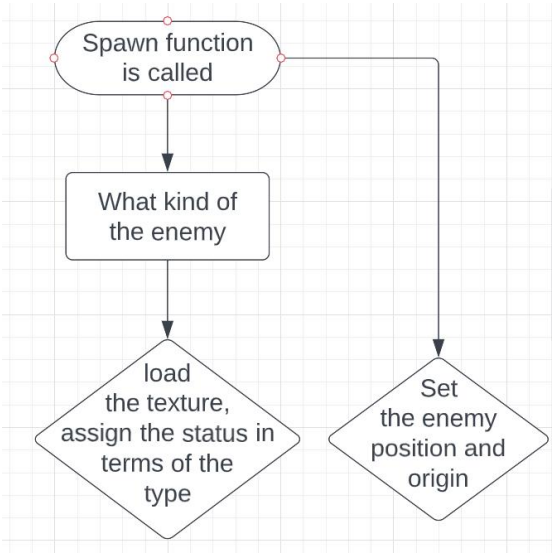
```
void Enemy::Chasing(float dt,Vector2f playerpos)
{
    if (type == 1 || type == 2) {
        if (playerpos.x > pos.x) {
            this->pos.x += status[1] * dt;
        }
        if (playerpos.x < pos.x) {
            this->pos.x -= status[1] * dt;
        }
        if (playerpos.y > pos.y) {
            this->pos.y += status[1] * dt;
        }
        if (playerpos.y < pos.y) {
            this->pos.y -= status[1] * dt;
        }
    }

    sprite.setPosition(pos);

    double angle = (atan2(playerpos.y - pos.y,playerpos.x - pos.x)* 180) / 3.141;
    sprite.setRotation((float)angle);

}
```



To make the enemy ready to be draw we have to set its texture, so we have

```
void Enemy::spawn(Vector2f pos, short type)
{
    switch (type)
    {
    case 1:
        //enemy map1
        texture.loadFromFile("enemy/level1.png");
        sprite.setTexture(texture);

        //status
        status[0] = normal_enemy1_health;
        status[1] = normal_enemy1_speed;
        status[2] = normal_enemy1_damage;
        break;

    case 2:
        //enemy map 2
        texture.loadFromFile("enemy/level2.png");
        sprite.setTexture(texture);

        //status
        status[0] = normal_enemy2_health;
        status[1] = normal_enemy2_speed;
        status[2] = normal_enemy2_damage;
        break;
```

```
    case 10:
        //enemy map 2
        texture.loadFromFile("boss/level2.png");
        sprite.setTexture(texture);

        //status
        status[0] = boss1_health;
        status[1] = 0;
        status[2] = boss1_damage;
        break;
    case 20:
        //enemy map 2
        texture.loadFromFile("boss/level2.png");
        sprite.setTexture(texture);

        //status
        status[0] = boss2_health;
        status[1] = 0;
        status[2] = boss2_damage;
        break;
    }

    this->pos.x = pos.x;
    this->pos.y = pos.y;

    sprite.setPosition(pos);
    sprite.setOrigin((float)texture.getSize().x / (unsigned)2, (float)texture.getSize().y / (unsigned)2);

}
```



4.Bullet

A bullet, shoot by the player , whose direction controlled by the player, when it hit the enemy it deal damage which is the damage of the player. It can fly in 8 direction.

```
class Bullet : public Animation
{
    Vector2f pos; // current position of the bullet

    bool inflight;// is the bullet flying or not
    float flightspeed;// bullet's speed

    float speedX;//bullet's speed on x axis
    float speedY;//bullet's speed on y axis

    //A border - maximum range that the bullet can fly away
    Vector2f minDistance;
    Vector2f maxDistance;

public:
    //Constructor
    Bullet();
    //Stop the bullet when it collide or touch its border
    void stop();
    //is the bullet still in flight
    bool isinflight();
    //shoot the bullet
    void shoot(Player player);
    //get the globalbounds for collision
    FloatRect getPos();
    //Update the bullet
    void Flying(float dt);
    //Draw the bullet
    void draw(RenderWindow& window);
};
```

So the idea is draw the bullet at the player's current position,then it fly away till hit the enemy or over its max range and to check the collision we need bullet's global bounds so we have

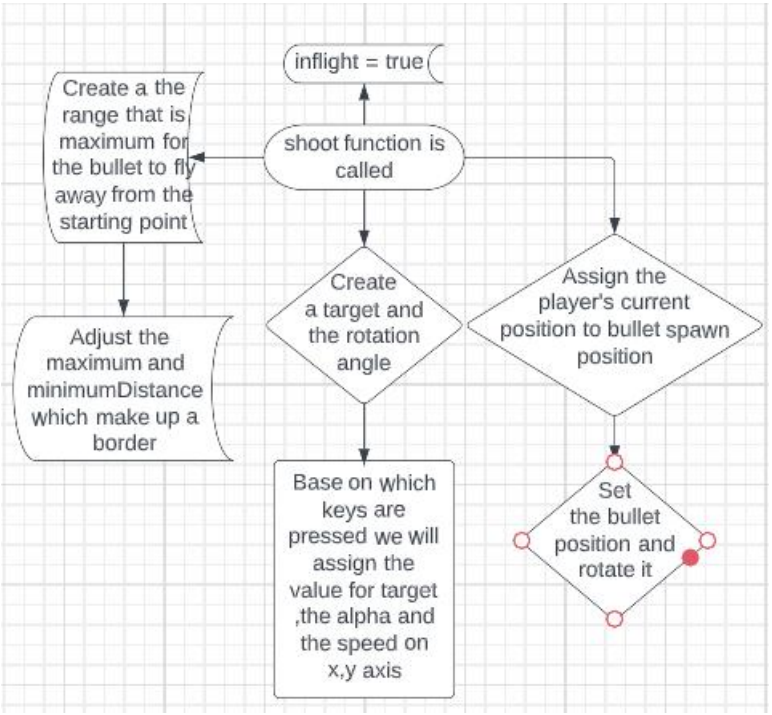
- getPos() functions : to get its sprite global bounds
-stop() : to stop the bullet by changing its fly status
- isinFlight() : to know is the bullet flying or not
- draw() : draw the bullet to the window

As we had discuss about the starting point of the bullet and how its flight, here we have shoot() function:

```
void Bullet::shoot(Player player)
{
    float alpha = 0;
    Vector2f target = Vector2f(0,0);
    if (Keyboard::isKeyPressed(Keyboard::Left)) {
        target.x = player.getpos().x - 1000;
        speedX = flightspeed;
        alpha = 180 ;
    }
    if (Keyboard::isKeyPressed(Keyboard::Right)) {
        target.x = player.getpos().x + 1000;
        speedX = flightspeed;
    }
    if (Keyboard::isKeyPressed(Keyboard::Up)) {
        target.y = player.getpos().y - 1000;
        speedY = flightspeed;
        alpha = 270;
    }
    if(Keyboard::isKeyPressed(Keyboard::Down)) {
        target.y = player.getpos().y + 1000;
        speedY = flightspeed;
        alpha = alpha + 90;
    }
    if (Keyboard::isKeyPressed(Keyboard::Down)&&Keyboard::isKeyPressed(Keyboard::Left)){
        alpha = 135;
    }
    if (Keyboard::isKeyPressed(Keyboard::Down)&&Keyboard::isKeyPressed(Keyboard::Right)){
        alpha = 45;
    }
    if (Keyboard::isKeyPressed(Keyboard::Up)&&Keyboard::isKeyPressed(Keyboard::Left)){
        alpha = -135;
    }
    if (Keyboard::isKeyPressed(Keyboard::Up)&&Keyboard::isKeyPressed(Keyboard::Right)){
        alpha = -45;
    }
    inflight = true;

    this->pos.x = player.getpos().x;
    this->pos.y = player.getpos().y;
```

```
    this->pos.x = player.getpos().x;
    this->pos.y = player.getpos().y;

    //Position the bullet ready to be drawn
    sprite.setPosition(pos);
    sprite.setRotation(alpha);

    //Set the speed on x and y axis to make the bullet shoot smoothly

    // Point the bullet in the right direction
    if (target.x < pos.x)
    {
        speedX *= -1;
    }
    if (target.y < pos.y)
    {
        speedY *= -1;
    }

    //Set a max range of 1000 pixels
    float range = 1000;
    minDistance.x = pos.x - range;
    minDistance.y = pos.y - range;

    maxDistance.x = pos.x + range;
    maxDistance.y = pos.y + range;
}
```

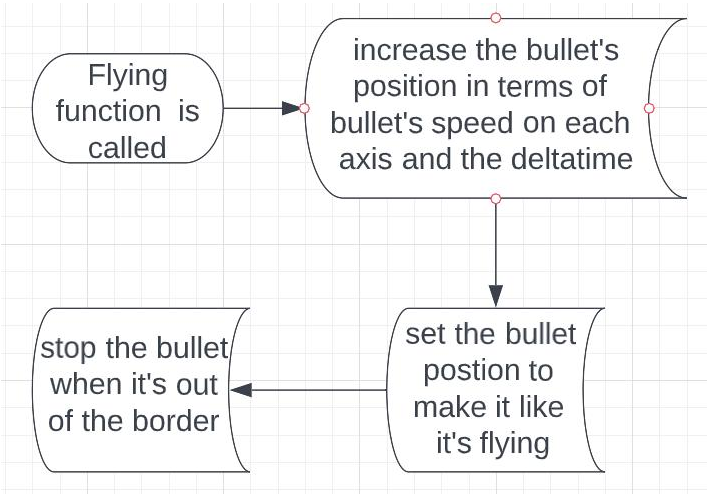
So first we create a vector2f which will be adjust to make up a fake target base on the keypress and a float angle to rotate the sprite later.



As we discussed, the bullet will stop when it hit the enemy or it out of range, mean it fly out of its border.

```
void Bullet::Flying(float dt)
{
    //Update the pos
    this->pos.x += speedX * dt;
    this->pos.y += speedY * dt;

    sprite.setPosition(pos);

    if (pos.x < minDistance.x || pos.x > maxDistance.x || pos.y < minDistance.y || pos.y > maxDistance.y) {
        inflight = false;
    }

}
```

When we set inflight to false the draw function won't work, so on other side it mean stop the bullet



5. Pickups

To make the game playable, we decide to put on pickups which will heal our player.

```
class Pickup : public Animation
{
private:

    //Status
    short Status;
    //Spawning and disappearing
    bool Spawned;               // is the pickups spawned
    float SecondsSinceSpawn;    // The moment when the pick up is spawned
    float SecondsSinceDeSpawn;  // The moment when the pick up disappear
    float SecondsToLive;        // How long is pickup appear on the window once
    float SecondsToWait;        // How long will the pick up appear again after it disappear
public:
    //Constructor
    Pickup();
    //Prepare the pickup
    void spawn(RenderWindow& window);
    //Check the pos of a pickup
    FloatRect getPos();
    //Draw to the window
    void draw(RenderWindow& window);
    //Update
    void update(float dt, RenderWindow& window);
    //When the player pick it up
    short picked_up();
    //Check if it spawned
    bool isSpawned();
};
```

Here we have a constructor, in this class constructor is important because the value of status of the pickup is fixed over the time .

```
Pickup::Pickup()
{
    texture.loadFromFile("Pickups/health.png");
    this->Status = 30;

    sprite.setTexture(texture);
    sprite.setOrigin(texture.getSize().x/2.0f, texture.getSize().y / 2.0f);
    sprite.setScale(5, 5);

    SecondsToLive = START_SECONDS_TO_LIVE;
    SecondsToWait = START_WAIT_TIME;
    SecondsSinceDeSpawn = 0;
    SecondsSinceSpawn = 0;
    Spawned = false;
}
```

Not very different from other, in constructor we just assign all of the variable. Here we

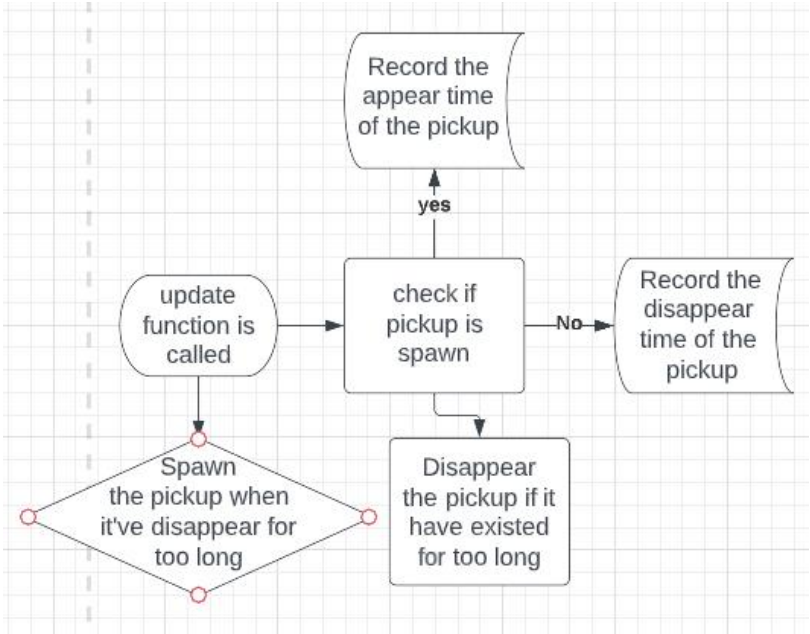
- load the texture then set the texture to the sprite
- set the origin and set scale

- set all the appear and disappear related variables.

```cpp
void Pickup::spawn(RenderWindow& window)
{
    Spawned = true;

    srand((int)time(NULL));
    //Random locatiom
    float x = rand() % (window.getSize().x);
    float y = rand() % (window.getSize().y);

    SecondsSinceSpawn = 0;

    sprite.setPosition(x, y);
}
```

We will assign true to spawned, then set the position of the bullet randomly and set it.

```cpp
void Pickup::update(float dt,RenderWindow& window)
{
    if (Spawned == true)
    {
        SecondsSinceSpawn += dt;
    }
    else
    {
        SecondsSinceDeSpawn += dt;
    }
    // Do we need to hide a pickup?
    if (SecondsSinceSpawn > SecondsToLive && Spawned == true)
    {
        // Remove the pickup and put it somewhere else
        Spawned = false;
        SecondsSinceDeSpawn = 0;
    }
    // Do we need to spawn a pickup
    if (SecondsSinceDeSpawn > SecondsToWait && !Spawned)
    {// spawn the pickup and reset the timer
        spawn(window);
    }
}
```



When the player collide with the pickup, we will all this function, first it will assign false to spawned which will make the pickup disappear.

```
short Pickup::picked_up()
{
    SecondsSinceDeSpawn = 0;

    Spawned = false;

    return this->Status;
}
```

6. Skill

Boss must differ from enemy, it can't not just chase the enemy and deal dame by it. So we have skill class, which give the boss ability to deal dame.

```
class Skill : public Animation{
private:

    //Spawning and disappearing
    bool Spawned;               // is the pickups spawned
    float Cast_time;            // How long does the skill need to repair before spawn
    float Appear_time;          // How long does the skill appear
    float temp;                 // To measure how long the game has run
public:
    Skill(int type);                                                    //Default constructor
    void Cast();                                                // cast is for the player
    void update(float dt,float gametime,Vector2f pos,Vector2f target);      // for boss
    void draw(RenderWindow& window);                                    //Draw
    FloatRect getPos();                                                 //get the blobal bounds of the skill
    bool isSpawned();                                                   //Check if the skill is casted
};
```

The way skill is written is very similar to how to write pickup. Here we have

- spawned : to check if it's spawned yet
- Cast_time : the amount of time before a skill is spawned
- Appear_time : how long can a skill appear on window
- temp : to store the value that can be use to evaluate when to spawned and disappear the skill

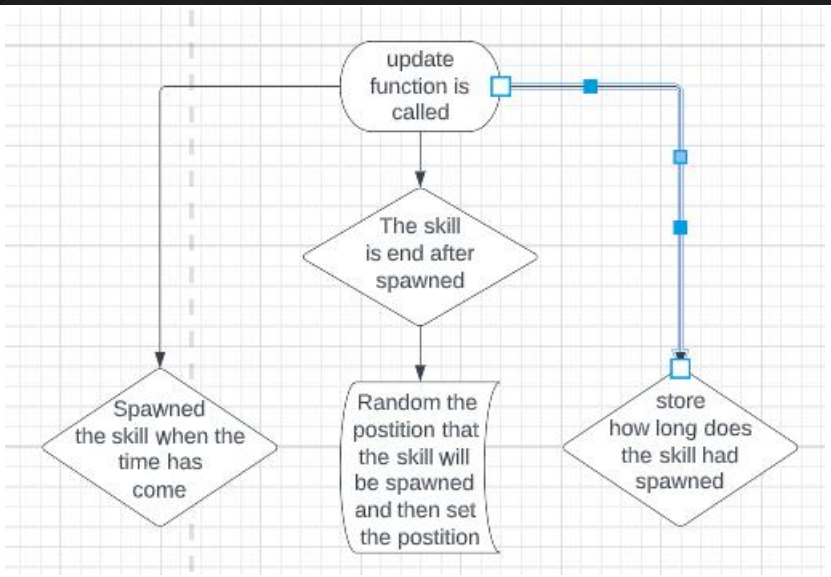
And :

- A default contructor which hold the texture and variable to spawned a skill from Boss

```
Skill::Skill(int type)
{
    if (type == 10) {
        texture.loadFromFile("Skills/level1.png");
    }
    else {
        texture.loadFromFile("Skills/level2.png");
    }
    sprite.setTexture(texture);
    sprite.setOrigin(texture.getSize().x / 2.0f, texture.getSize().y / 2.0f);

    Cast_time = 1.0f;
    Appear_time = 0.5f;
    temp = 0;
    Spawned = false;
}
```

- Update function to spawn the skill from the Boss:

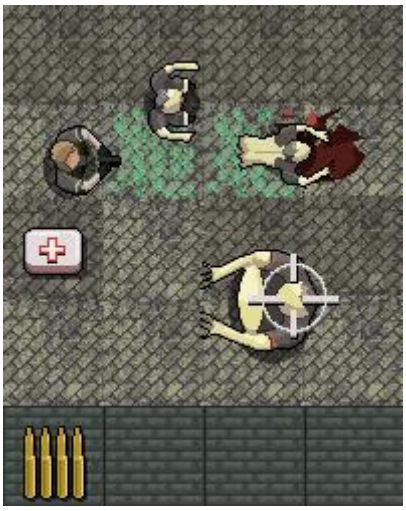
```
void Skill::update(float dt,float gametime, Vector2f pos,Vector2f target)
{

    if (Spawned == true)
    {
        temp +=dt;
    }

    if (gametime > Cast_time)
    {
        Spawned = true;
        Cast_time += Cast_time;
    }

    if (temp > Appear_time && Spawned == true) {
        temp = 0;
        Spawned = false;
    }
    Vector2f pos1; pos1.x = 0; pos1.y = 0;
    srand(gametime);

    if (pos.x > target.x) {
        pos1.x = pos.x - (float)(rand() % 200) ;
    }
    else
        pos1.x = pos.x + (float)(rand() % 200);

    if (pos.y > target.y) {
        pos1.y = pos.y - (float)(rand() % 200);
    }
    else
        pos1.y = pos.y + (float)(rand() % 200);

    sprite.setPosition(pos1);

    //Update(dt);
}
```



7.Map Generator

There is an easy or idle way to pass in a background as can be call a map by choosing random image in the internet, then creating a Sprite and Pass the Texture. Finally, with ” window.draw()” it will appear as a boring image behind your’s contents.

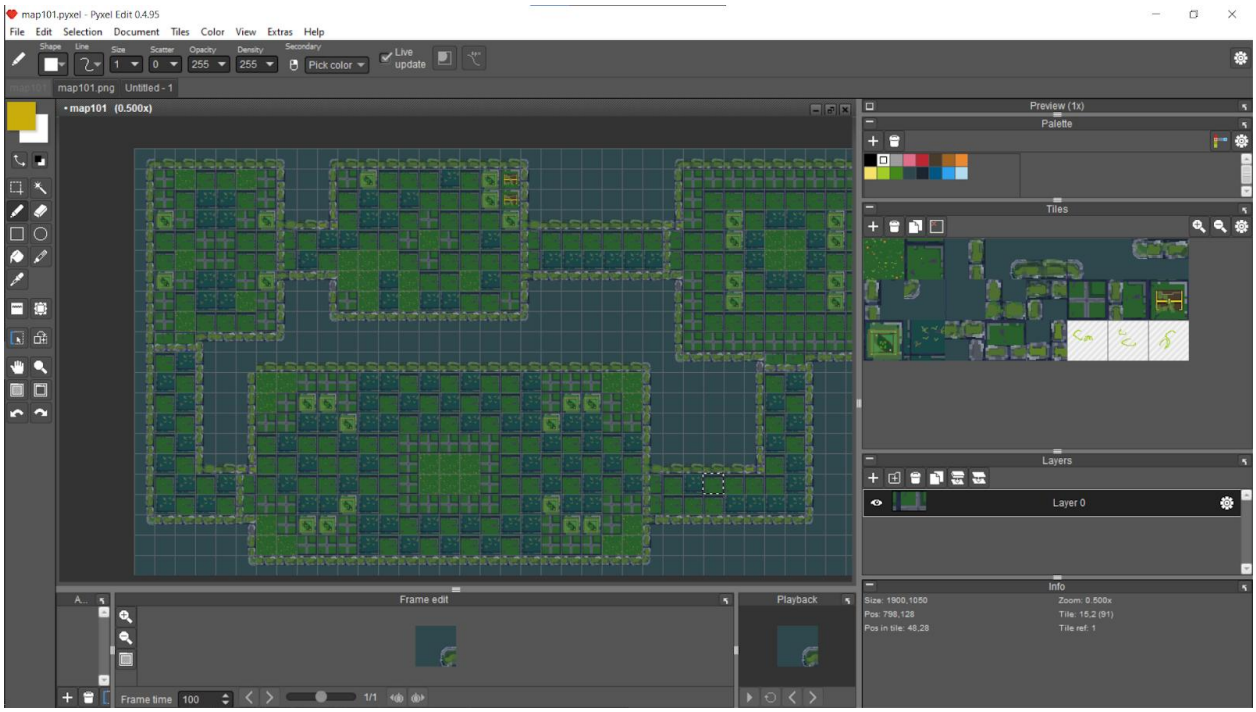
We do not want that. At first i use Vertex Array which is suggested in the book, but then a problem appears since we want our map to be seperated to many “space” and want to perform a Dungeon games with variety of shape playground not just a boring Arena, which is just a whole window screen.



We decide to use vector to draw a map form a Text file. The mechanism is pretty simple. First, we need to have a tileset which is the block with the size of which we choose to be 50.px by 50.px. We use a drawing software call Pyxel.edit to perform preparation:

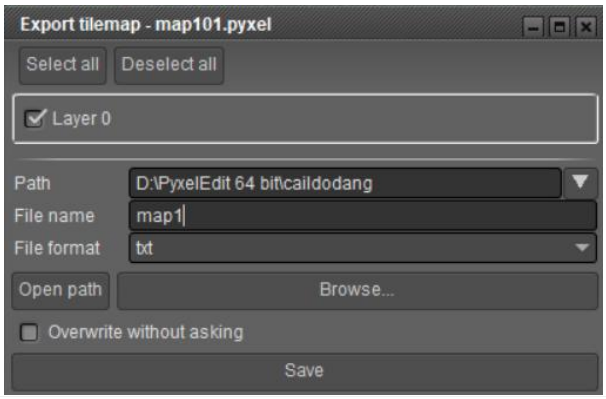
(tile set, map1)

(tile set map2)



(creating map using Pyxel.Edit)

After creating a map, the Software can export the tilemap as a txt file which contains the sets of tiles in used in left
to right order.




```
tileswide 38
tileshigh 21
tilewidth 50
tileheight 50

layer 0
3,4,4,4,4,4,4,5,10,3,4,4,4,4,4,4,4,4,5,10,10,10,10,10,10,3,4,4,4,4,4,4,4,4,4,5,
2,13,0,1,1,0,13,8,10,2,13,16,1,1,1,17,1,16,15,8,10,10,10,10,10,10,2,13,13,13,13,13,13,13,13,13,8,
2,13,1,17,17,1,13,8,10,2,1,17,1,17,1,1,1,16,15,8,10,10,10,10,10,10,2,13,1,1,1,1,1,1,1,1,13,8,
2,16,13,17,17,13,16,11,4,20,1,17,17,1,13,1,1,1,16,11,4,4,4,4,4,4,20,13,1,16,1,17,17,1,16,1,13,8,
2,1,1,13,13,1,1,14,1,17,17,1,1,13,0,13,1,17,1,14,1,1,1,1,1,1,14,1,1,16,17,0,0,17,16,1,13,8,
2,1,0,13,13,1,1,14,17,1,0,0,0,1,13,1,1,1,1,14,17,17,17,17,17,17,14,1,1,1,17,0,0,17,1,1,13,8,
2,16,13,17,17,13,16,12,7,18,0,0,17,0,1,1,0,17,1,12,7,7,7,7,7,18,13,1,16,1,17,17,1,16,1,13,8,
2,13,1,17,17,0,13,8,10,2,13,0,17,0,17,17,1,1,13,8,10,10,10,10,10,10,2,13,1,16,1,1,1,1,16,1,13,8,
2,13,0,1,1,1,13,8,10,6,7,7,7,7,7,7,7,7,9,10,10,10,10,10,10,2,13,1,1,1,1,1,1,1,1,13,8,
2,19,19,12,7,7,7,9,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,2,13,13,13,13,13,13,13,13,13,13,8,
2,1,17,8,10,10,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,5,6,7,7,7,18,19,19,12,7,7,7,9,
2,17,1,8,10,2,0,0,13,13,13,17,17,1,17,1,17,1,1,17,13,13,13,0,0,8,10,10,10,10,2,1,17,8,10,10,10,10,
2,1,17,8,10,2,0,13,16,16,13,17,1,17,1,17,1,17,1,17,13,16,16,13,0,8,10,10,10,10,2,17,1,8,10,10,10,10,
2,17,1,8,10,2,13,13,17,17,16,17,17,1,17,1,17,1,17,17,16,17,17,13,13,8,10,10,10,10,2,1,17,8,10,10,10,10,
2,1,17,8,10,2,1,17,1,1,17,1,1,13,13,13,13,13,1,1,17,1,17,1,17,8,10,10,10,10,2,17,1,8,10,10,10,10,
2,17,1,11,4,20,1,1,17,1,1,17,1,13,0,0,0,13,1,17,1,17,1,17,1,11,4,4,4,4,20,1,17,8,10,10,10,10,
2,1,17,1,17,14,1,17,1,1,17,1,1,13,0,0,0,13,1,1,17,1,17,1,17,14,1,17,1,1,1,1,17,8,10,10,10,10,
2,17,1,17,1,14,13,13,17,17,16,17,1,13,13,13,13,13,1,17,16,17,17,13,13,14,1,1,1,17,17,17,17,8,10,10,10,10,
6,7,7,7,7,18,0,13,16,16,13,17,17,1,17,1,17,1,17,17,13,16,16,13,0,12,7,7,7,7,7,7,9,10,10,10,10,
10,10,10,10,10,2,0,0,13,13,13,17,1,17,1,17,1,17,1,17,13,13,13,0,0,8,10,10,10,10,10,10,10,10,10,10,10,
10,10,10,10,10,6,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,9,10,10,10,10,10,10,10,10,10,10,10,
```

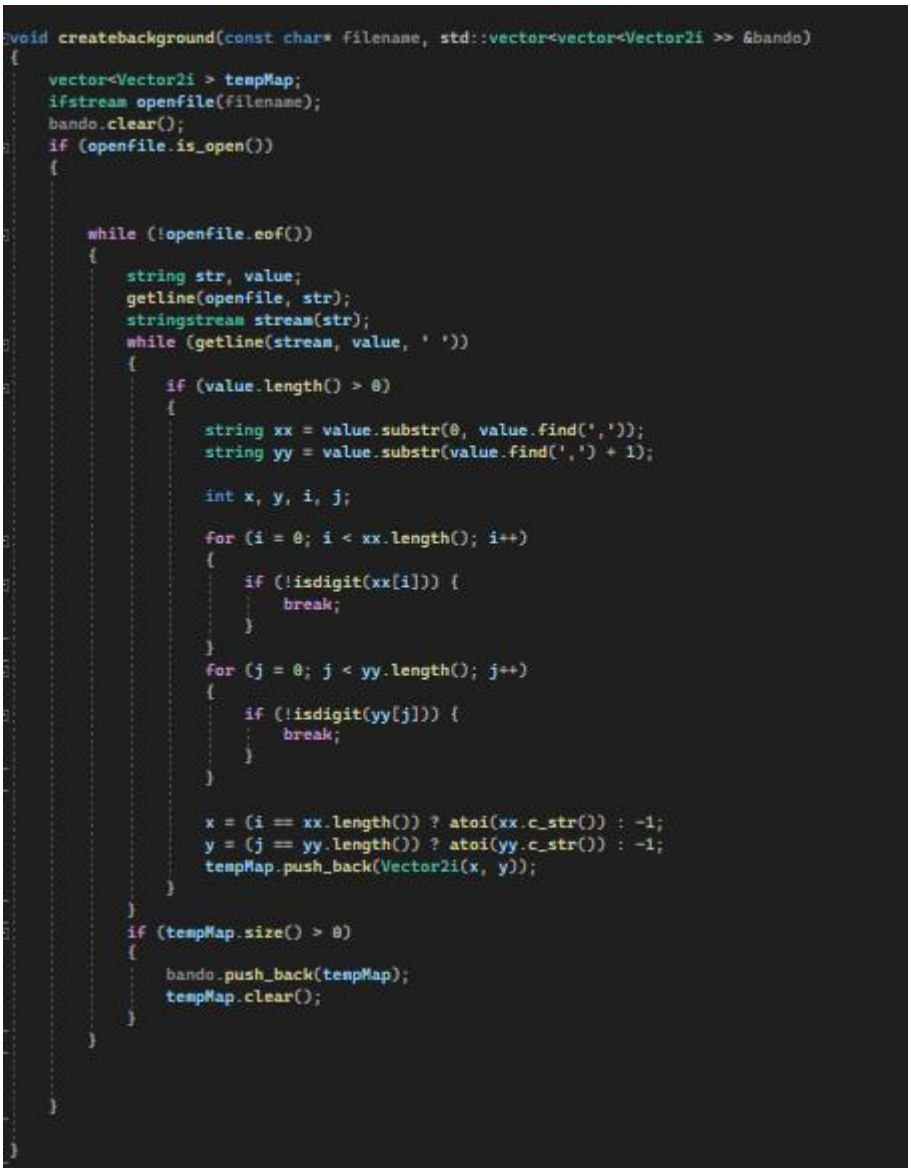
(the result TXT file)

```
3,0 4,0 4,0 4,0 4,0 4,0 4,0 4,0 4,0 4,0 5,0 2,1 2,1 2,1 2,1 2,1 2,1 2,1 2,1 3,0 4,0 4,0 4,0 4,0 4,0 4,0 4,0 4,0 4,0 4,0 4,0 4,0 4,0 4,0 4,0 4,0 5,0
2,0 7,1 7,1 0,2 5,1 5,1 5,1 5,1 5,1 5,1 5,1 0,1 2,1 2,1 2,1 2,1 2,1 2,1 2,1 2,1 2,0 5,1 5,1 5,1 5,1 5,1 5,1 5,1 5,1 5,1 5,1 5,1 5,1 5,1 5,1 5,1 0,1
2,0 0,2 0,2 1,0 1,2 1,0 1,2 1,0 1,0 5,1 0,1 2,1 2,1 2,1 2,1 2,1 2,1 2,1 2,1 2,0 5,1 1,0 1,0 1,0 1,0 1,0 1,0 1,2 1,2 1,2 1,0 1,0 1,0 1,0 1,0 1,0 0,1
2,0 0,2 1,2 1,0 1,0 1,0 1,0 1,2 1,0 5,1 3,1 4,0 4,0 4,0 4,0 5,0 2,1 2,1 2,1 2,0 5,1 1,2 0,2 0,2 1,0 1,0 1,0 1,2 1,0 1,0 1,2 1,2 1,0 0,2 0,2 1,0 5,1 0,1
2,0 5,1 1,2 1,2 1,2 1,2 1,2 1,0 1,0 1,0 6,1 1,0 1,0 1,0 0,1 2,1 2,1 2,1 2,0 5,1 1,2 1,0 1,0 1,0 1,0 1,0 1,0 1,0 1,0 1,2 1,2 1,2 1,0 5,1 0,1
2,0 5,1 1,0 1,0 1,2 1,2 1,2 1,0 1,0 1,2 1,0 6,1 1,0 1,2 1,0 0,1 2,1 2,1 2,1 2,0 5,1 1,2 1,0 1,2 1,2 1,2 1,2 0,2 1,2 1,2 0,2 1,0 1,0 1,0 1,0 1,0 5,1 0,1
2,0 5,1 1,2 1,0 1,0 1,0 1,0 1,2 1,0 0,2 4,1 7,0 2,2 1,0 1,0 0,1 2,1 2,1 2,1 2,0 5,1 1,2 1,0 1,2 1,2 1,2 1,0 1,0 1,0 1,0 1,2 1,2 1,2 1,0 1,0 1,0 5,1 0,1
2,0 5,1 1,0 1,2 1,0 1,0 1,2 1,0 0,2 0,2 0,1 2,1 2,0 1,0 1,2 0,1 2,1 2,1 2,1 2,0 5,1 1,2 1,0 1,0 1,2 1,2 1,2 0,2 1,0 1,0 0,2 1,0 1,2 1,0 1,2 1,0 5,1 0,1
2,0 5,1 5,1 5,1 5,1 5,1 0,2 0,2 7,1 7,1 0,1 2,1 2,0 1,2 1,0 0,1 2,1 2,1 2,1 2,0 5,1 1,0 0,2 0,2 1,0 1,0 1,0 1,0 1,0 1,0 1,0 1,0 1,2 1,0 1,2 1,0 5,1 0,1
6,0 7,0 7,0 7,0 7,0 7,0 7,0 7,0 7,0 7,0 1,1 2,1 2,0 1,0 1,2 0,1 2,1 2,1 2,1 2,0 5,1 1,0 1,0 1,0 1,0 1,0 1,0 1,0 1,0 1,0 1,2 1,2 1,0 0,2 1,0 5,1 0,1
2,1 2,1 2,1 2,1 2,1 2,1 2,1 2,1 2,1 2,1 2,1 2,1 2,1 2,0 1,0 1,0 0,1 2,1 2,1 2,1 2,0 5,1 5,1 5,1 5,1 5,1 5,1 5,1 5,1 5,1 5,1 1,2 1,2 1,0 1,0 0,2 1,0 5,1 0,1
2,1 2,1 2,1 2,1 2,1 2,1 2,1 2,1 2,1 2,1 2,1 2,1 2,1 2,0 3,2 3,2 0,1 2,1 2,1 2,1 6,0 7,0 7,0 7,0 7,0 7,0 7,0 7,0 2,2 5,1 1,2 1,2 1,2 1,0 0,2 1,2 5,1 0,1
2,1 3,0 4,0 4,0 4,0 4,0 4,0 4,0 4,0 4,0 4,0 4,0 4,2 1,2 1,2 3,1 4,0 4,0 4,0 4,0 4,0 4,0 4,0 4,0 5,0 2,1 2,1 2,1 2,0 5,1 1,0 1,0 1,2 1,0 0,2 1,0 5,1 0,1
2,1 2,0 7,1 5,1 0,2 5,1 5,1 5,1 5,1 5,1 5,1 5,1 5,1 1,0 1,0 5,1 5,1 5,1 5,1 0,2 5,1 5,1 5,1 5,1 0,1 2,1 2,1 2,1 2,0 5,1 1,0 1,0 1,2 1,2 0,2 1,0 5,1 0,1
2,1 2,0 0,2 0,2 1,2 1,0 1,2 1,0 1,0 0,2 0,2 1,0 1,0 1,0 1,0 1,0 1,0 1,0 0,2 1,0 1,0 1,2 1,2 5,1 3,1 4,0 4,0 4,0 4,2 5,1 1,0 1,0 1,0 1,2 0,2 1,0 5,1 0,1
2,1 2,0 5,1 1,2 0,2 1,0 1,0 1,0 1,0 0,2 0,2 1,2 1,2 1,2 1,0 1,0 1,0 0,2 1,0 1,2 1,0 1,0 1,2 5,1 6,1 1,0 1,0 1,0 1,0 1,0 1,0 1,0 1,0 1,2 0,2 1,0 5,1 0,1
2,1 2,0 5,1 1,2 1,2 1,2 1,2 1,2 1,2 1,0 1,0 1,0 1,0 1,0 1,0 1,0 1,0 1,0 1,2 1,0 1,0 1,2 1,2 6,1 1,0 1,0 1,0 1,0 1,0 1,2 1,2 1,2 1,0 1,0 1,2 5,1 0,1
2,1 2,0 5,1 1,2 1,2 0,2 1,2 1,0 1,0 1,0 1,0 1,0 1,0 1,0 1,0 1,0 1,0 1,0 1,0 1,0 1,2 1,2 1,2 4,1 7,0 7,0 7,0 2,2 5,1 1,0 1,0 1,0 1,2 0,2 0,2 5,1 0,1
2,1 2,0 5,1 1,2 0,2 1,2 1,2 1,2 1,0 1,2 1,0 1,2 1,2 1,2 1,2 1,0 1,2 0,2 1,0 1,2 1,2 1,2 0,2 0,2 0,1 2,1 2,1 2,1 2,0 5,1 1,0 1,0 1,2 1,2 1,0 0,2 5,1 0,1
2,1 2,0 5,1 0,2 5,1 5,1 5,1 5,1 5,1 5,1 5,1 5,1 5,1 5,1 5,1 5,1 5,1 5,1 0,2 5,1 5,1 0,2 7,1 7,1 0,1 2,1 2,1 2,1 2,0 5,1 5,1 5,1 5,1 5,1 5,1 5,1 0,1
2,1 6,0 7,0 7,0 7,0 7,0 7,0 7,0 7,0 7,0 7,0 7,0 7,0 7,0 7,0 7,0 7,0 7,0 7,0 7,0 7,0 7,0 7,0 7,0 7,0 1,1 2,1 2,1 2,1 6,0 7,0 7,0 7,0 7,0 7,0 7,0 7,0 1,1
```

(After we modified)

The code algorithms:

Each of the tile in the tiles set has different value of x and y as seen above.
In oder to store which value I use **vector<vector<Vector2i** data type. We need to store the value in 2d so we initial a vector2d as (vector<vector), then the value is integer and each of the members need to store two value as the tile in position (x,y) so we use Vector2i to store it. And to store each member as a Vector2i we create a tempMap as **vector<Vector2i.**



```
void createbackground(const char* filename, std::vector<vector<Vector2i >> &bando)
{
    vector<Vector2i > tempMap;
    ifstream openfile(filename);
    bando.clear();
    if (openfile.is_open())
    {


        while (!openfile.eof())
        {
            string str, value;
            getline(openfile, str);
            stringstream stream(str);
            while (getline(stream, value, ' '))
            {
                if (value.length() > 0)
                {
                    string xx = value.substr(0, value.find(','));
                    string yy = value.substr(value.find(',') + 1);

                    int x, y, i, j;

                    for (i = 0; i < xx.length(); i++)
                    {
                        if (!isdigit(xx[i])) {
                            break;
                        }
                    }
                    for (j = 0; j < yy.length(); j++)
                    {
                        if (!isdigit(yy[j])) {
                            break;
                        }
                    }

                    x = (i == xx.length()) ? atoi(xx.c_str()) : -1;
                    y = (j == yy.length()) ? atoi(yy.c_str()) : -1;
                    tempMap.push_back(Vector2i(x, y));
                }
            }
            if (tempMap.size() > 0)
            {
                bando.push_back(tempMap);
                tempMap.clear();
            }
        }


    }
}
```

(the map loading function)

+While (not the end of the file) we create a temp str just to collect the value directly from left to right in the Txt file getline(openfile,str) . Then i pass it into the stringstream call stream because the value can be a very long number not just a single character as 1 or 2.

+While (getline(stream, value, ‘ ‘)) is to get the value in the stream and pass it to the Value separate by the ‘ ‘. Then we pass the string xx to the values as given above before the ‘ , ‘ dectectd but for the left of the ‘ , ‘. Similarly for the value in the right hand side of the ‘ , ‘ we store it in the yy string using the short form of if else function : (*variable* = (*condition*) ? *expressionTrue* : *expressionFalse)*

Then we pass each pair of x and y to the tempMap vector

Then we use two for loop as two collect the digit type value, cause sometimes I want to pass a non digit value as ‘x’ to specify in another use.
Then after going through the two string xx and yy we pass the value and use the function call **atoi** to store it in the integerdata type. After that we pass each pair of x and y to the tempMap vector by pushBack(), and then use the same method to pass the vector<Vector 2i temMap to the 2d vector. Then we clear() the value in temp map and load till the end of the file is reached.

```
//Draw the map1
for (int i = 0; i < map1.size(); i++)
{
    for (int j = 0; j < map1[i].size(); j++)
    {
        if (map1[i][j].x != -1 && map1[i][j].y != -1)
        {
            tile1.setPosition(j * 50, i * 50);
            tile1.setTextureRect(IntRect(map1[i][j].x * 50, map1[i][j].y * 50, 50, 50));
            window.draw(tile1);

        }
    }
}
```

Finally we use 2 for loop to loop through the vector 2d map, set the tile position as i*50 and j*50 because we want to set the tile continuously and the tilesize is 50. Then we setTextureRect as two specify the characteristic of the tile which are left, right, top and bottom. Then draw it using window.draw().

The collision Map is nearly the same method. The difference is that we want to store the (Pass through) or (Not Passthrough) as 1 and 0, so we just need a vector<vector<int as each member are either 1 or 0.

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 0 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 1 1 0 0 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 1 1 1 1 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1
1 0 0 0 0 0 1 1 0 0 1 1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 0 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 0 0 1
1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 0 0 1
1 1 1 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 0 0 1
1 1 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1
1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1
1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 0 1
1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 1
1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
void setcoli(const char* filename, std::vector<vector<int >>& colmap)
{
    vector<int > tempMap;
    ifstream openfile(filename);
    colmap.clear();
    if (openfile.is_open())
    {
        while (!openfile.eof())
        {
            string str, value;
            getline(openfile, str);
            stringstream stream(str);
            while (getline(stream, value, ' '))
            {
                if (value.length() > 0)
                {
                    int a = atoi(value.c_str());
                    tempMap.push_back(a);


                }
            }

            colmap.push_back(tempMap);
            tempMap.clear();
        }


    }

}
```

Similar to the map but this time we get the single int value.

```
// Draw the colision map
for (int i = 0; i < colmap1.size(); i++)
{
    for (int j = 0; j < colmap1[i].size(); j++)
    {
        if (colmap1[i][j] == 1)
        {
            Vector2f pos(j * 50, i * 50);
            coli.setPosition(pos);
            window.draw(coli);
            if (player1.getRectPos().intersects(coli.getGlobalBounds()))
            {
                player1.update(player1, dt.asSeconds());
            }


        }

    }
```

Then Draw it and using the intersects method

8.Main function

Of course we have to include all of the header file

```
#include <iostream>
#include "Player.h"
#include "Enemy.h"
#include <cstdlib>
#include "Bullet.h"
#include "Pickup.h"
#include "Skill.h"
#include "Animation.h"
#include "background.h"

using namespace std;
using namespace sf;
```

So before starting the game, enemy is not just one only, it has to be a horde, so we create a non member function of enemy called Horde to create many many enemies:

```
//Create enemies horde
Enemy* horde(int amount, int type) {
    Enemy* enemies = new Enemy[amount];

    for (int i = 0; i < amount; i++) {
        srand(int(time(NULL)) * i);

        Vector2f pos;
        pos.x = (float)(rand() % 801 )+ 200;
        pos.y = (float)(rand() % 600 )+ 100;

        enemies[i].spawn(pos, type);
    }
    return enemies;
}
```

In this function, we need the type of the enemy to spawn the right horde, then we random two float number to put in the enemy position to spawn them randomly.

Starting in the main function, to play game we need a window and in roughlike type, we need to keep the window focus on the player so we need View

```
//window and view
RenderWindow window(VideoMode(920, 800), "Finding Memo", Style::Resize | Style::Close);
View view(Vector2f(viewwidth, viewheight), Vector2f(viewwidth, viewheight));
```

Then we create the player and time variables to know how long had the game run.

```
//Create the player
Player player1;

Time gametime,dashtimer,movetimer;
Clock clock;
```

We create a horde of enemy by make it as a array and then allocate it by called horde function

```
////////////////////////////Create horde//////////////////////////
int numenemies = 10, aliveenemies = numenemies;
Enemy* enemies = nullptr;
delete[] enemies;
enemies = horde(numenemies, 1);
////////////////////////////////////////////////////////////////////
```

Then we create the Boss and spawn it

```
/////////////////Create Boss//////////////////////
Enemy Boss;
Boss.spawn(Vector2f(600.0f, 600.0f), 10);
float dtime=0;
////////////////////////////////////////////////
```

So many enemy, then we need so many bullet to kill them all

```
//////////////////////////Bullet//////////////////
int numberbullet = 40;
Bullet* bullet = new Bullet[40];
Time lastPressed;
int i1 = 0;
float firerate = 1;
////////////////////////////////////////////////
```

Create the pickup to heal the player and the Boss's skill

```
//////////////////////some pickup/////////////////////
Pickup health;
//////////////////////skill/////////////////////////
Skill boss1(10);
```

Finally, the map

```
////////////////////////////////////Map/////////////////////////////////////
vector<vector<Vector2i >> map1;
Sprite tile1;
Texture tileTexture1;
tileTexture1.loadFromFile("baihatcuatoi.png");
tile1.setTexture(tileTexture1);

vector<vector<int >> colmap1;

createbackground("tinhyeuhoctro.txt", map1);
//set colision
setcoli("colisiontinhyeu.txt", colmap1);

RectangleShape coli (Vector2f(50, 50));
////////////////////////////////////////////////////////////////////////////
```

So we have finish reparing all the stuff the play the game so just create a loop to keep loading and updating them

```
//Game loop
while (window.isOpen())
{
    //Deltatime and gametotaltime
    Time dt = clock.restart();
    gametime += dt;
    dtime = dt.asSeconds();

    //Skills.setPosition(player1.getpos());
    alert.setPosition(Vector2f(player1.getpos().x, player1.getpos().y - 50));

    // size up the health bar
    healthBar.setSize(Vector2f(player1.get_health() / 3.0f, 10.0f));
    healthBar.setOrigin(healthBar.getSize().x / 2.0f, healthBar.getSize().y / 2.0f);
    healthBar.setPosition(player1.getpos()+Vector2f(0.0f,-30.0f));

    Event ev;
```

Here we use window.isOpen() to create the gameloop

Create and assign the dt to clock which mean it store the time for each times the game loop finish. Assign the gametime to store the whole procession' s time.

Create the HealthBar to let us is us about to die, and an event variable to know what is happen to the window.

We set the view and use the event variable to control the window.

```
//Event polling
while (window.pollEvent(ev))
{
    switch (ev.type)
    {
    case Event::Closed:
        window.close();
        break;
    case Event::Resized:
        ResizeView(window, view);
        break;
    case Event::KeyPressed:
        if (ev.key.code == Keyboard::Escape)
            window.close();
        break;
    case Event::TextEntered:
        if (ev.text.unicode < 128) {};
        break;
    }
}

//set the viewcentre
view.setCenter(player1.getpos());
```

```
//Make the view resizalbe
void ResizeView(RenderWindow& window, View& view) {
    float ratio = float(window.getSize().x) / float(window.getSize().y);
    view.setSize(viewheight * ratio, viewheight);
}
```

Then we update all the thing, bullet, enemies, boss, player and pickups

```
//Fire the bullet
if (Keyboard::isKeyPressed(Keyboard::C)) {
    if (gametime.asMilliseconds() - lastPressed.asMilliseconds() > 500 / firerate) {
        //shoot the bullet
        if (i1 > numberbullet)
            i1 = 0;

        bullet[i1].shoot(player1);
        lastPressed = gametime;
        i1++;
    }
}

//Update the bullet
for (int i = 0; i < 40; i++) {
    bullet[i].Flying(dt.asSeconds());
}

//Update the pickup
health.update(dt.asSeconds(),window);

//Update the skill
boss1.update(dtime,gametime.asSeconds(), Boss.getpos(),player1.getpos());

//Update the player
player1.Moving(dt.asSeconds(),window);
```

```
//Let Enemies chase the player
if (Boss.dead() == false) {
    //Boss.Chasing(dt.asSeconds(), player1.getpos());

    if (player1.getpos().x > Boss.getpos().x) {
        pos1.x += 40 * dtime;
    }
    if (player1.getpos().x < Boss.getpos().x) {
        pos1.x -= 40 * dtime;
    }
    if (player1.getpos().y > Boss.getpos().y) {
        pos1.y += 40 * dtime;
    }
    if (player1.getpos().y < Boss.getpos().y) {
        pos1.y -= 40 * dtime;
    }
    Boss.move(pos1.x, pos1.y);
    pos1.x = 0; pos1.y = 0;
}

//Update the enemies
for (int i = 0; i < numenemies; i++) {
    if (enemies[i].dead() == false) {
        enemies[i].Chasing(dt.asSeconds(), player1.getpos());
    }
}
```

```
if (Keyboard::isKeyPressed(Keyboard::Z)) {
    if (gametime.asMilliseconds() - dashtimer.asMilliseconds() > 500) {

        if (Keyboard::isKeyPressed(Keyboard::Left)) {
            pos.x -= 100*player1.get_speed()*dt.asSeconds();
            cout << pos.x <<"left"<<endl;
        }
        if (Keyboard::isKeyPressed(Keyboard::Right)) {
            pos.x += 100 * player1.get_speed() * dt.asSeconds();
            cout << pos.x << "right" << endl;
        }
        if (Keyboard::isKeyPressed(Keyboard::Up)) {
            pos.y -= 100 * player1.get_speed() * dt.asSeconds();
            cout << pos.y << "up" << endl;
        }
        if (Keyboard::isKeyPressed(Keyboard::Down)) {
            pos.y += 100 * player1.get_speed() * dt.asSeconds();
            cout << pos.y << "down" << endl;
        }

        dashtimer = gametime;
        player1.move(pos);
        pos.x = 0; pos.y = 0;
    }
}
```

Give the player ability to dash

And now we draw

```
// Draw the colision map
for (int i = 0; i < colmap1.size(); i++)
{
    for (int j = 0; j < colmap1[i].size(); j++)
    {
        if (colmap1[i][j] == 1)
        {
            Vector2f pos((float)(j * 50),(float)(i * 50));
            coli.setPosition(pos);
            window.draw(coli);
            if (player1.getRectPos().intersects(coli.getGlobalBounds()))
            {
                player1.update(player1, dt.asSeconds());
            }
        }
    }
}
```

We draw the map which will collide with the player and a normal map lie all over it

```
//Draw the map1
for (int i = 0; i < map1.size(); i++)
{
    for (int j = 0; j < map1[i].size(); j++)
    {
        if (map1[i][j].x != -1 && map1[i][j].y != -1)
        {
            tile1.setPosition((float)(j * 50),(float)(i * 50));
            tile1.setTextureRect(IntRect(map1[i][j].x * 50, map1[i][j].y * 50, 50, 50));
            window.draw(tile1);
        }
    }
}
```

Next we draw the enemies and the bullet

```
//Draw the enemies
for (int i = 0; i < numenemies; i++)
{
    if (enemies[i].dead() == false) {
        enemies[i].draw(window);
    }
}

if (Boss.dead() == false)
    Boss.draw(window);

//Draw the bullet
for (int i = 0; i < 40; i++) {
    if (bullet[i].isinflight() == true) {
        bullet[i].draw(window);
    }
}
```

Draw the pickup and the player

```
//Draw the pickups
if (health.isSpawned() == true)
{
    health.draw(window);
}

if (boss1.isSpawned() == true) {
    boss1.draw(window);
}

player1.draw(window);
```

Then we check the collision by function intersect with globalbounds

Enemies and the bullet

```
//Bullet and enemies
for (int i = 0; i < 40; i++)
{
    for (int j = 0; j < numenemies; j++)
    {
        if (bullet[i].isinflight() && enemies[j].dead() == false)
        {
            if (bullet[i].getPos().intersects(enemies[j].getRectPos()))
            {
                // Stop the bullet
                bullet[i].stop();
                // Register the hit and see if it was a kill
                enemies[j].getHit(player1.get_damage());
            }

        }
    }
}
```

Boss and Bullet

```
for (int i = 0; i < 40; i++) {
    if (bullet[i].isinflight() && Boss.dead() == false) {
        if (bullet[i].getPos().intersects(Boss.getRectPos()))
        {
            // Stop the bullet
            bullet[i].stop();
            // Register the hit and see if it was a kill
            Boss.getHit(player1.get_damage());
        }
    }
}
```

To make the enemies not collapse with each other we check the collide and mvoe them

```
//Enemies and Enemies
for (int i = 0; i < numenemies; i++)
{
    for (int j = i; j < numenemies; j++)
    {
        if (enemies[i].dead() == false && enemies[j].dead() == false) {
            if (enemies[i].getRectPos().intersects(enemies[j].getRectPos())) {
                if (enemies[i].getpos().x > enemies[j].getpos().x) {
                    enemies[i].move(1, 0);
                    enemies[j].move(-1, 0);
                }
                if (enemies[i].getpos().x < enemies[j].getpos().x) {
                    enemies[i].move(-1, 0);
                    enemies[j].move(1, 0);
                }
                if (enemies[i].getpos().y > enemies[j].getpos().y) {
                    enemies[i].move(0, 1);
                    enemies[j].move(0, -1);
                }
                if (enemies[i].getpos().y < enemies[j].getpos().y) {
                    enemies[i].move(0, -1);
                    enemies[j].move(0, 1);
                }
            }
        }
    }
}
```

Finally, pickup, skill with the player

```
//Player and the pickups
if (player1.getRectPos().intersects(health.getPos()) && health.isSpawned())
{
    player1.pick_up(health.picked_up());

}
//Skill and the player
if (player1.getRectPos().intersects(boss1.getPos()) && boss1.isSpawned())
{
    player1.hit(gametime,50);

}
```

At the end, we draw the healthBar, set the view and display all the stuff
But notice, the game will end when the player is dead so we need to close the window

```
window.draw(healthBar);
window.setView(view);
window.display();
//Close
if (player1.isdead() == true) {
    window.close();
}
```

And that is how I make a game with sfml and c++

IV. Reference

1. https://www.youtube.com/watch?v=axIgxBQVBg0&list=PL21OsoBLPpMOO6zyVlxZ4S4hwkY_SLRW9&ab_channel=HilzeVonck
2. https://github.com/Kofybrek/Super-Mario-Bros
3. Beginning C++ Game Programming