COCOS2D
C++

Game Project

By : Yassir Tali

2022/2023 - FSTT

In order to develop a 2d game with cocos2d :

We will :

- Create sprites
- Create the physics for the sprite
- Create the movement of map
- Move the player
- Create an if statement to detecting collisions
- Replace the scene with win or lose after collision
- Add the buttons with their functions
- Endless ground rolling

- Score increase
- Keyboard listener
- Update function that gonna track the game while running

To make :

- Level1 :

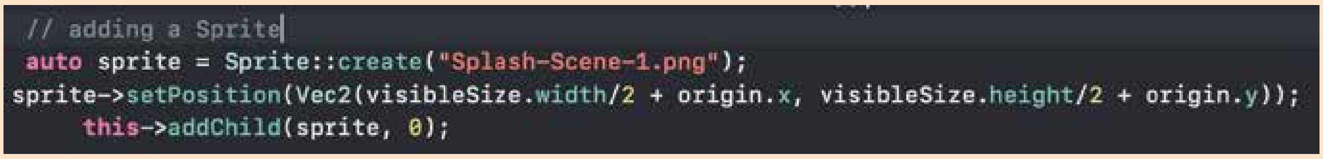
- Level2 :

- Level3 :

- Level4 :

Creating sprites

We create a sprite by using cocos2d ready class “Sprite ::Create”, then we set its position and we add it to te the current scene by using “ this->addChild” .

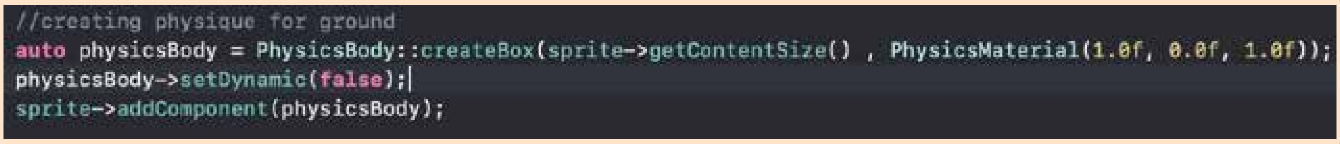


```
 // adding a Sprite
 auto sprite = Sprite::create("Splash-Scene-1.png");
sprite->setPosition(Vec2(visibleSize.width/2 + origin.x, visibleSize.height/2 + origin.y));
     this->addChild(sprite, 0);
```

The second argument of the addchild method “0 ” stands for the Z Axe .

Creating the physics for the sprite

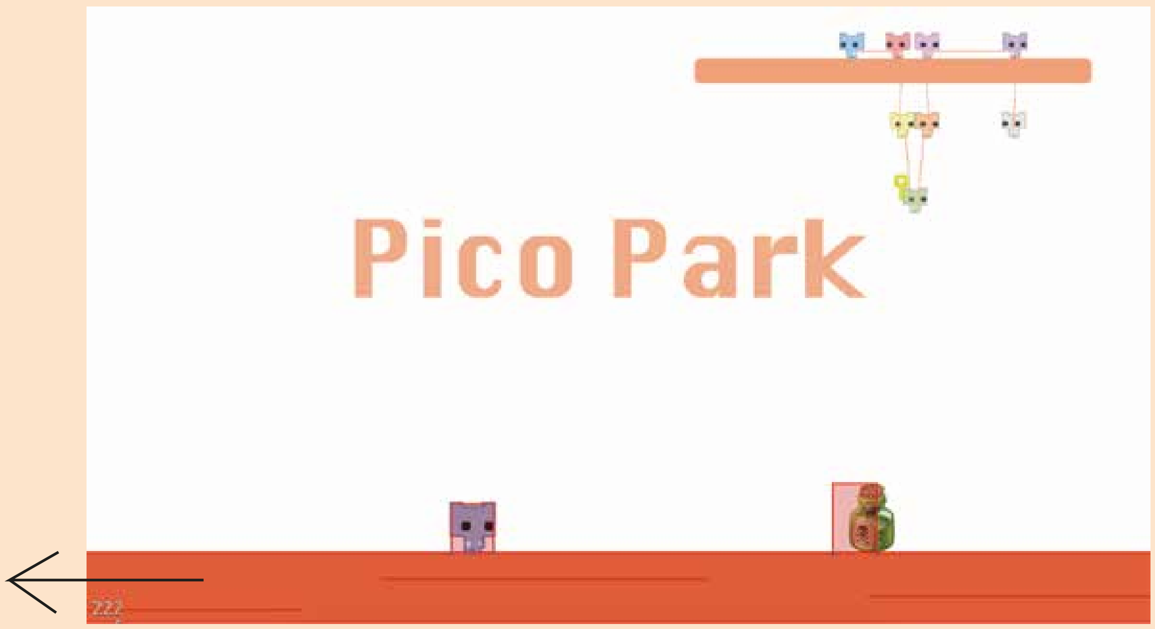
To create a physique, we start by getting the size from the sprite , we use another cocos2d ready class “PhysicsBody”. (We need to add Physics world to our scene in order to use this library).



```
//creating physique for ground
auto physicsBody = PhysicsBody::createBox(sprite->getContentSize() , PhysicsMaterial(1.0f, 0.0f, 1.0f));
physicsBody->setDynamic(false);
sprite->addComponent(physicsBody);
```

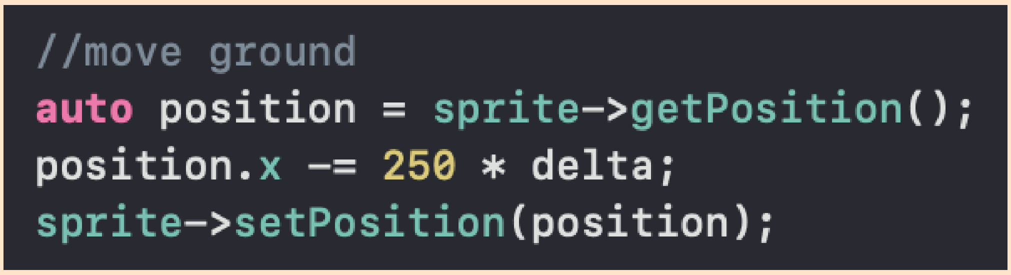
The second line is for desactivating the gravity and some physics rules.

Create the movement of map



The orange ground above is a 6000px width sprite. In the beginning the sprite is in it top left, after using a code, i make it move to the left so it shows like moving and the obstacles get generated in the right side by a function.
Finaly we got the effect of moving forward the map.

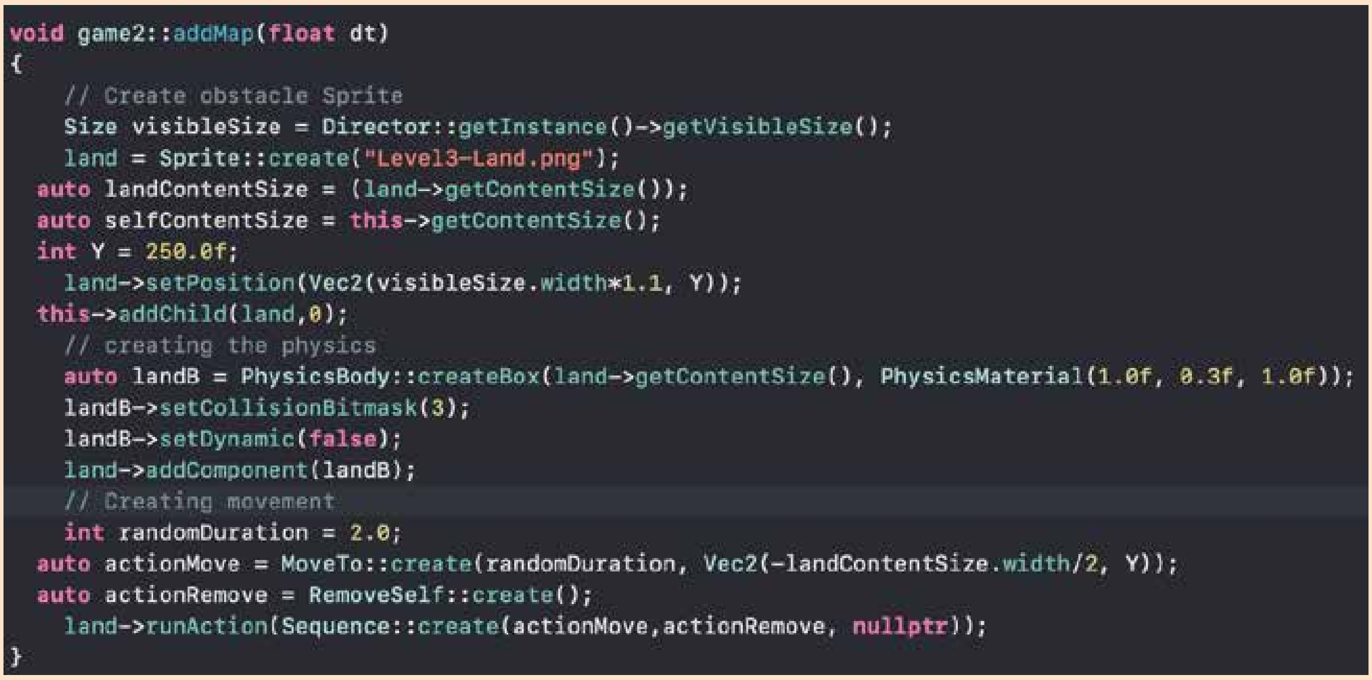
In order to move the ground, we first get the curent position, we make it move by using the second line of the above code, a logic simple code . We give back the new position every time (We put this code in update so it runs permently).



```
//move ground
auto position = sprite->getPosition();
position.x -= 250 * delta;
sprite->setPosition(position);
```

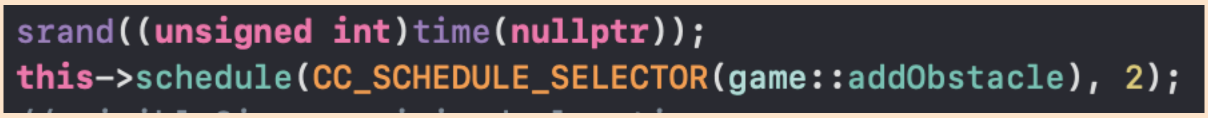
We use the same code for the win target, I just Set the win sprite position So Far so it takes a moment to come.

Using the same logic (Creating sprite -> physics -> moving it) We did a function that gonna create obstacles the right side of our scene and move it every time we call it (every part is commented bellow).



```
void game2::addMap(float dt)
{
    // Create obstacle Sprite
    Size visibleSize = Director::getInstance()->getVisibleSize();
    land = Sprite::create("Level3-Land.png");
  auto landContentSize = (land->getContentSize());
  auto selfContentSize = this->getContentSize();
  int Y = 250.0f;
    land->setPosition(Vec2(visibleSize.width*1.1, Y));
  this->addChild(land,0);
    // creating the physics
    auto landB = PhysicsBody::createBox(land->getContentSize(), PhysicsMaterial(1.0f, 0.3f, 1.0f));
    landB->setCollisionBitmask(3);
    landB->setDynamic(false);
    land->addComponent(landB);
    // Creating movement
    int randomDuration = 2.0;
  auto actionMove = MoveTo::create(randomDuration, Vec2(-landContentSize.width/2, Y));
  auto actionRemove = RemoveSelf::create();
    land->runAction(Sequence::create(actionMove,actionRemove, nullptr));
}
```

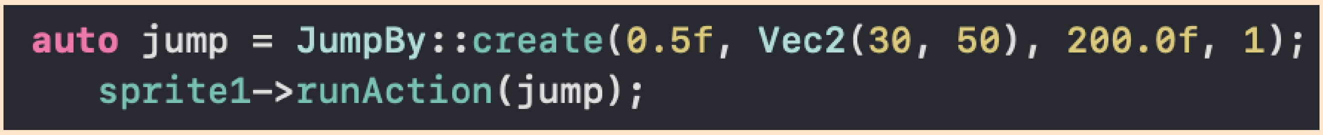
We call the function by using these two lines of code, Number 2 is the period where the function going to be called everytime.



```
srand((unsigned int)time(nullptr));
this->schedule(CC_SCHEDULE_SELECTOR(game::addObstacle), 2);
```

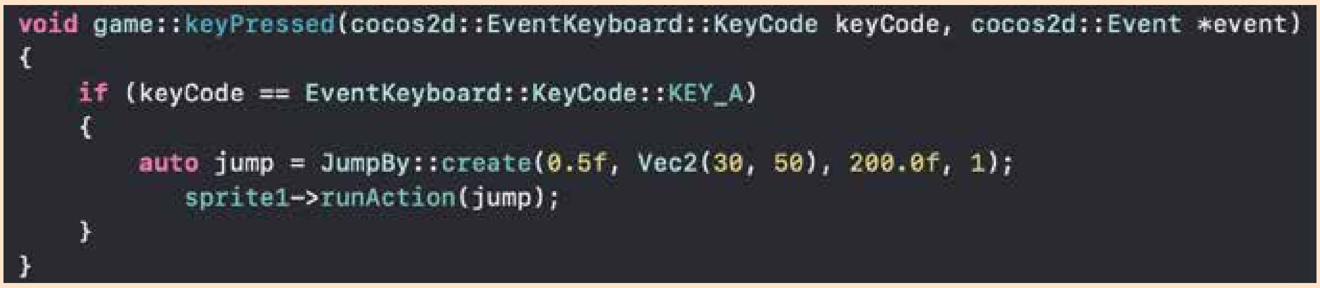
move the player

Previously, we used exesting function in the JumpBy class giving by Cocos2D.
So simple, just adding the details we want like shown below:



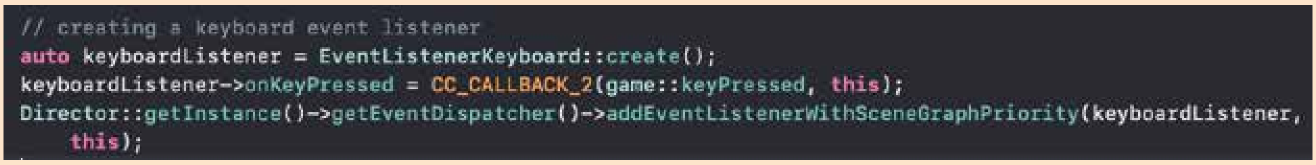
```
auto jump = JumpBy::create(0.5f, Vec2(30, 50), 200.0f, 1);
    sprite1->runAction(jump);
```

“0.5f” the time of jump, “vec2(30,50:)” the vectore of jump movement, “200.0f” is the height of jump, “1 ” the number of jumps on the period.



```
void game::keyPressed(cocos2d::EventKeyboard::KeyCode keyCode, cocos2d::Event *event)
{
    if (keyCode == EventKeyboard::KeyCode::KEY_A)
    {
        auto jump = JumpBy::create(0.5f, Vec2(30, 50), 200.0f, 1);
            sprite1->runAction(jump);
    }
}
```

We put the jump method in the function of keybord event so we can call it when we press a key button.

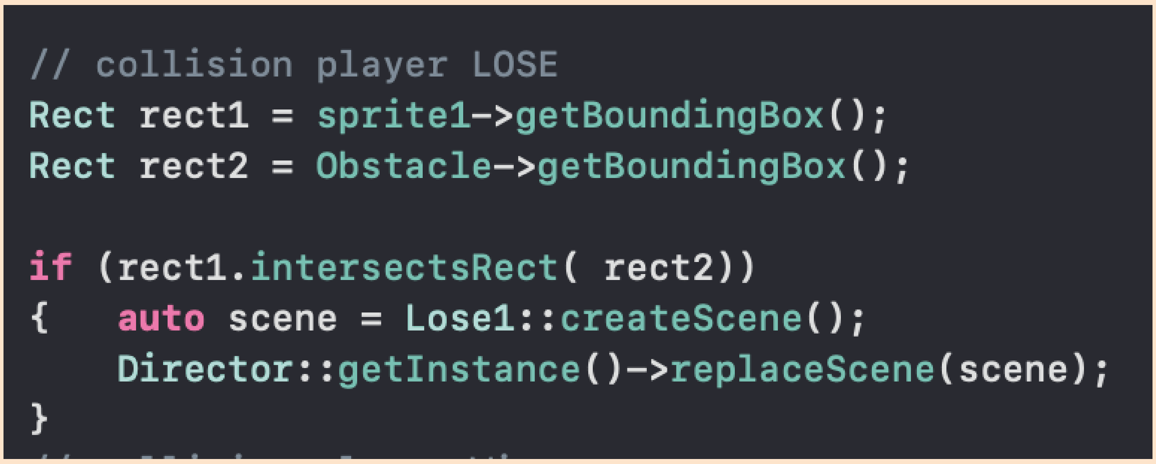


```cpp
// creating a keyboard event listener
auto keyboardListener = EventListenerKeyboard::create();
keyboardListener->onKeyPressed = CC_CALLBACK_2(game::keyPressed, this);
Director::getInstance()->getEventDispatcher()->addEventListenerWithSceneGraphPriority(keyboardListener,
    this);
```

We add this code in our bool init to declare the keyboard listener, other time we used ready cocos2d class.

Do X when colided

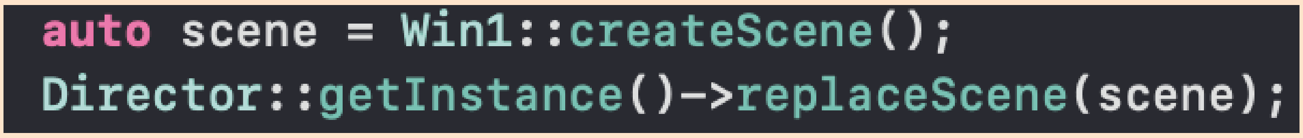
This code will replace the curent scene with Lose scene when there is a collision. Simple use of Cocos2d class and built in functions.
As we see the two elements (sprite1 and Obstacle) are the bodies we waiting their collisions. We put this code in update function.



```cpp
// collision player LOSE
Rect rect1 = sprite1->getBoundingBox();
Rect rect2 = Obstacle->getBoundingBox();

if (rect1.intersectsRect( rect2))
{   auto scene = Lose1::createScene();
    Director::getInstance()->replaceScene(scene);
}
```

Replacing the scene with win or lose after collision

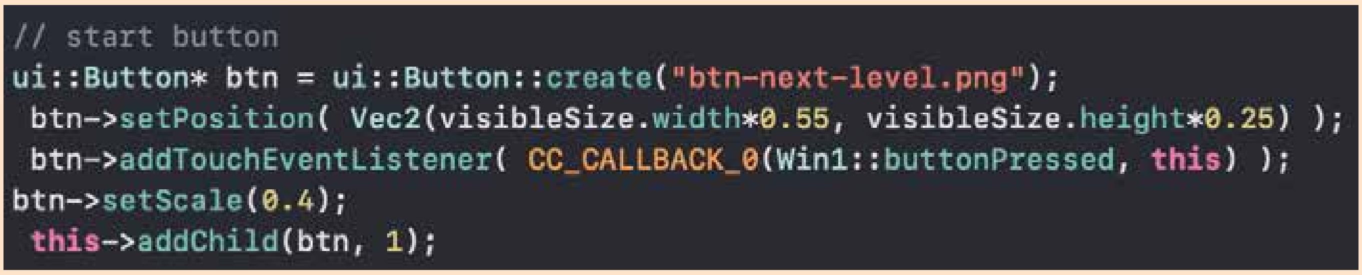
We will use the same code of the collision twice, one for lose other one for win. whenever we want to load a scene, we can put this two lines (replace “Win1 ” with the scene you want to go to):



```
auto scene = Win1::createScene();
Director::getInstance()->replaceScene(scene);
```

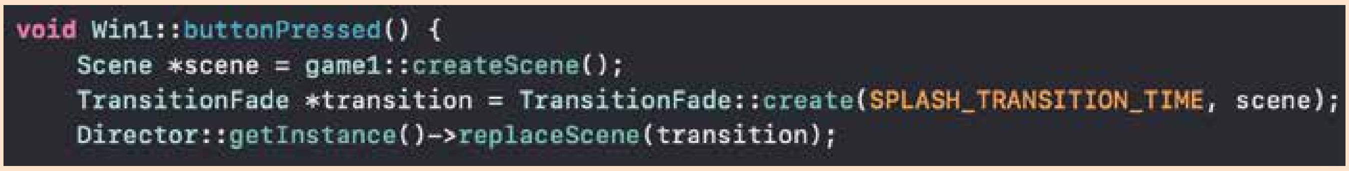
adding the buttons with their functions

We will use UI Cocos2d library. So first we need to call it Using (namespace ui ; #include "ui/CocosGUI.h").
After that lets add the code of the button.



```
// start button
ui::Button* btn = ui::Button::create("btn-next-level.png");
 btn->setPosition( Vec2(visibleSize.width*0.55, visibleSize.height*0.25) );
 btn->addTouchEventListener( CC_CALLBACK_0(Win1::buttonPressed, this) );
btn->setScale(0.4);
 this->addChild(btn, 1);
```

Its clear that all you need to do is to add the image of your button and the fucnction that you want call like i did below:

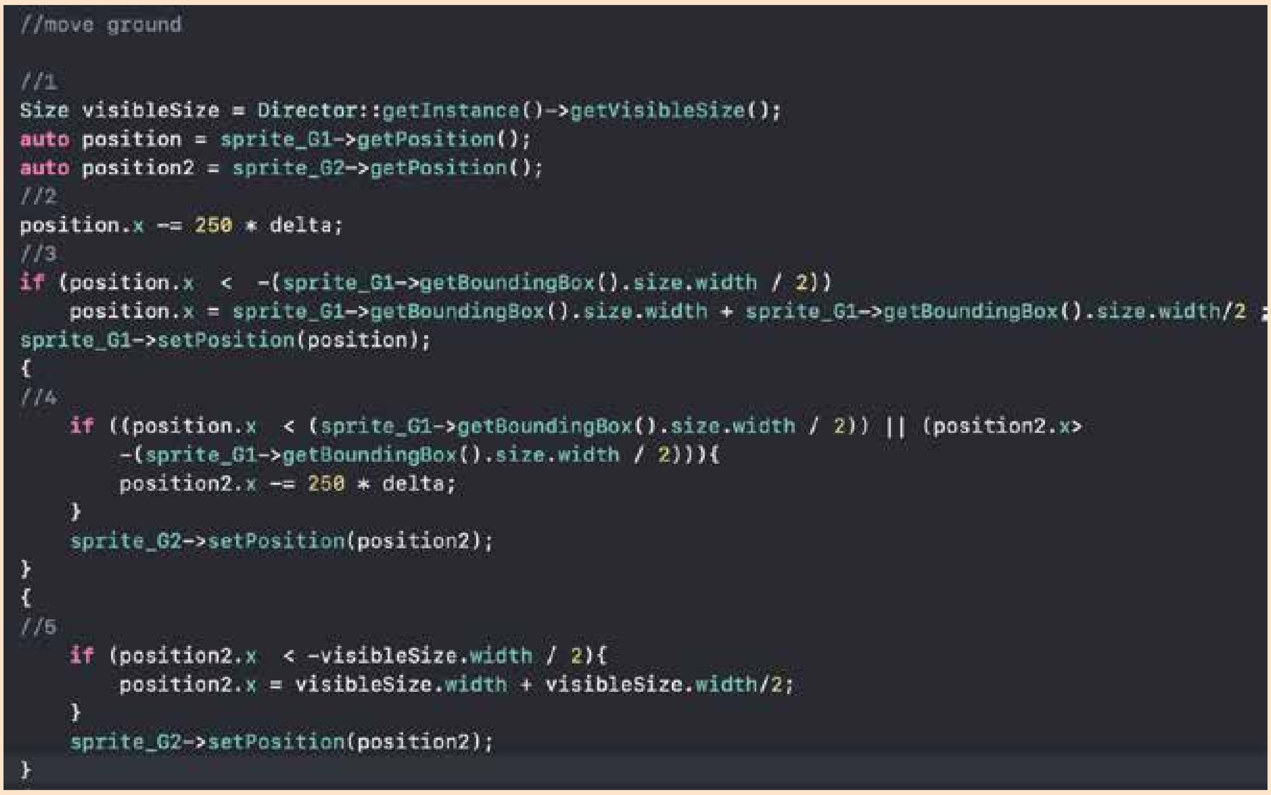


```
void Win1::buttonPressed() {
    Scene *scene = game1::createScene();
    TransitionFade *transition = TransitionFade::create(SPLASH_TRANSITION_TIME, scene);
    Director::getInstance()->replaceScene(transition);
```

This code is the same as the replacing scene. I just added a small transition.

endless rolling background

This Code takes me a bit of time. I've spent so much time trying methods from here and there but nothing worked for me for a unknow reason (maybe the difference of versions) . But anyway let explore my simple endless rolling ground.



```
//move ground

//1
Size visibleSize = Director::getInstance()->getVisibleSize();
auto position = sprite_G1->getPosition();
auto position2 = sprite_G2->getPosition();
//2
position.x -= 250 * delta;
//3
if (position.x  <  -(sprite_G1->getBoundingBox().size.width / 2))
    position.x = sprite_G1->getBoundingBox().size.width + sprite_G1->getBoundingBox().size.width/2 ;
sprite_G1->setPosition(position);
{
//4
    if ((position.x  < (sprite_G1->getBoundingBox().size.width / 2)) || (position2.x>
        -(sprite_G1->getBoundingBox().size.width / 2))){
        position2.x -= 250 * delta;
    }
    sprite_G2->setPosition(position2);
}
{
//5
    if (position2.x  < -visibleSize.width / 2){
        position2.x = visibleSize.width + visibleSize.width/2;
    }
    sprite_G2->setPosition(position2);
}
```

I explain the code above from section //1 to section //5 :

1 - Getting the position of the Two copies of my ground that going to create the rolling effect (one sprite goes left to show like the player move forward, the other sprite move directly after the first so there is no gap)
i put the initial position of the first in the middle of the screen, the second is directly outside the right side of my visible screen.

2 - A simple code to move the first ground sprite.

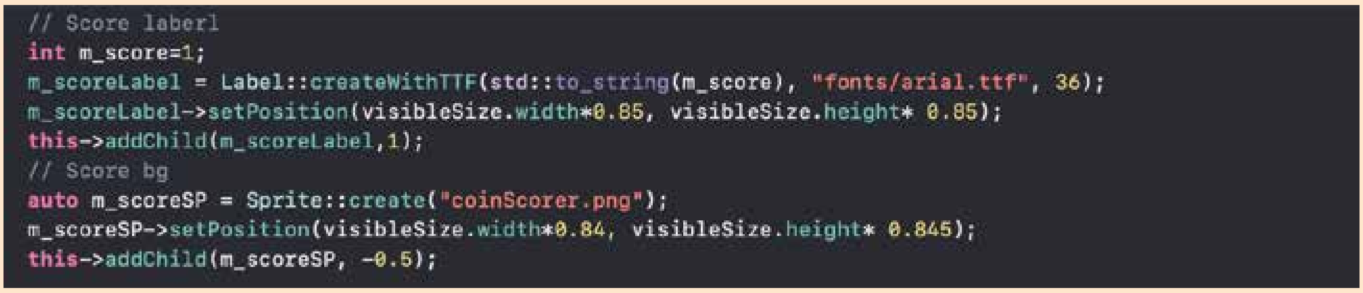
3 - when the first sprite getout 100% from the screen, reposition it to the start point.

4 - When the first sprite move to the left directly move the seconde one with the same speed .

5 - when the second sprtie is 100% out take back to his initial position.

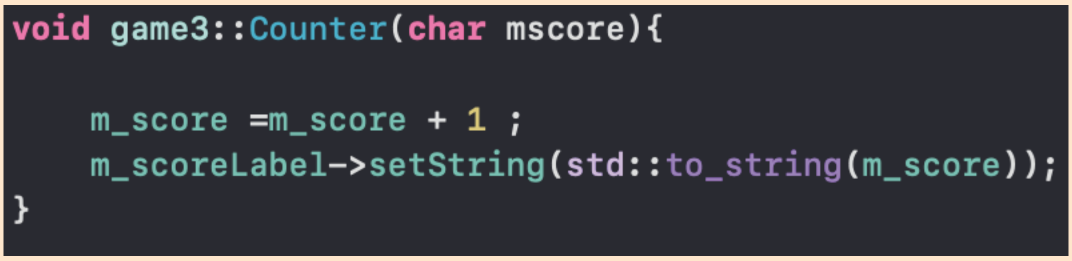
Score increase

First We need to create the label to show the Score with a good background:
Don't forget to add the signature of the sprites and the lables in the header file.



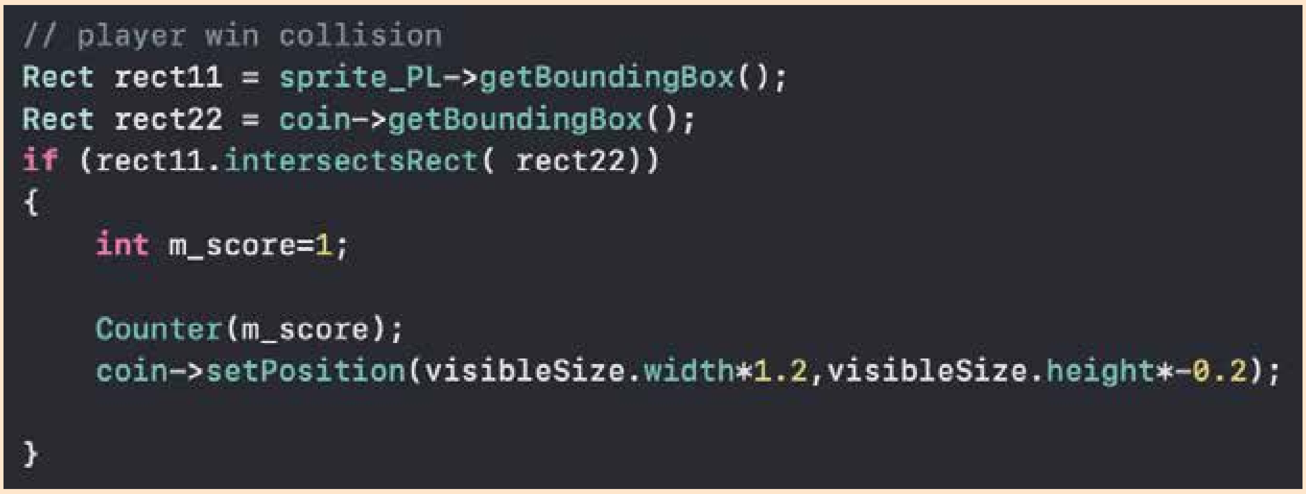
```
// Score laberl
int m_score=1;
m_scoreLabel = Label::createWithTTF(std::to_string(m_score), "fonts/arial.ttf", 36);
m_scoreLabel->setPosition(visibleSize.width*0.85, visibleSize.height* 0.85);
this->addChild(m_scoreLabel,1);
// Score bg
auto m_scoreSP = Sprite::create("coinScorer.png");
m_scoreSP->setPosition(visibleSize.width*0.84, visibleSize.height* 0.845);
this->addChild(m_scoreSP, -0.5);
```

Using a small function to increase the score when we call it from collision :



```
void game3::Counter(char mscore){

    m_score =m_score + 1 ;
    m_scoreLabel->setString(std::to_string(m_score));
}
```

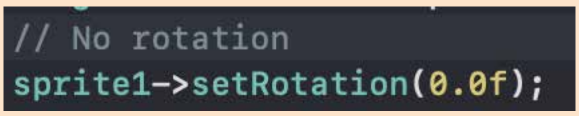
With no complication we can add it to our collision statement:



```
// player win collision
Rect rect11 = sprite_PL->getBoundingBox();
Rect rect22 = coin->getBoundingBox();
if (rect11.intersectsRect( rect22))
{
    int m_score=1;

    Counter(m_score);
    coin->setPosition(visibleSize.width*1.2,visibleSize.height*-0.2);

}
```

Update function

Well, I think it is important to mention that the update function is the place where we add all the actions that happen within the game.
Update function is a basic thing in all the game frameworks and engines as an example i will show you what i've did when i wanted to keep my player straight with no rotations :



```
// No rotation
sprite1->setRotation(0.0f);
```

The update will always take care of this instruction every frame/s:

LEVELS

I will keep it simple, no repetition i am going to underline the problems and solution that i've made + i will explain the simple mechanise of my levels.

LEVEL 1

There is no need to repeat the code that i have shown you before so i will mention the steps and add the special things :

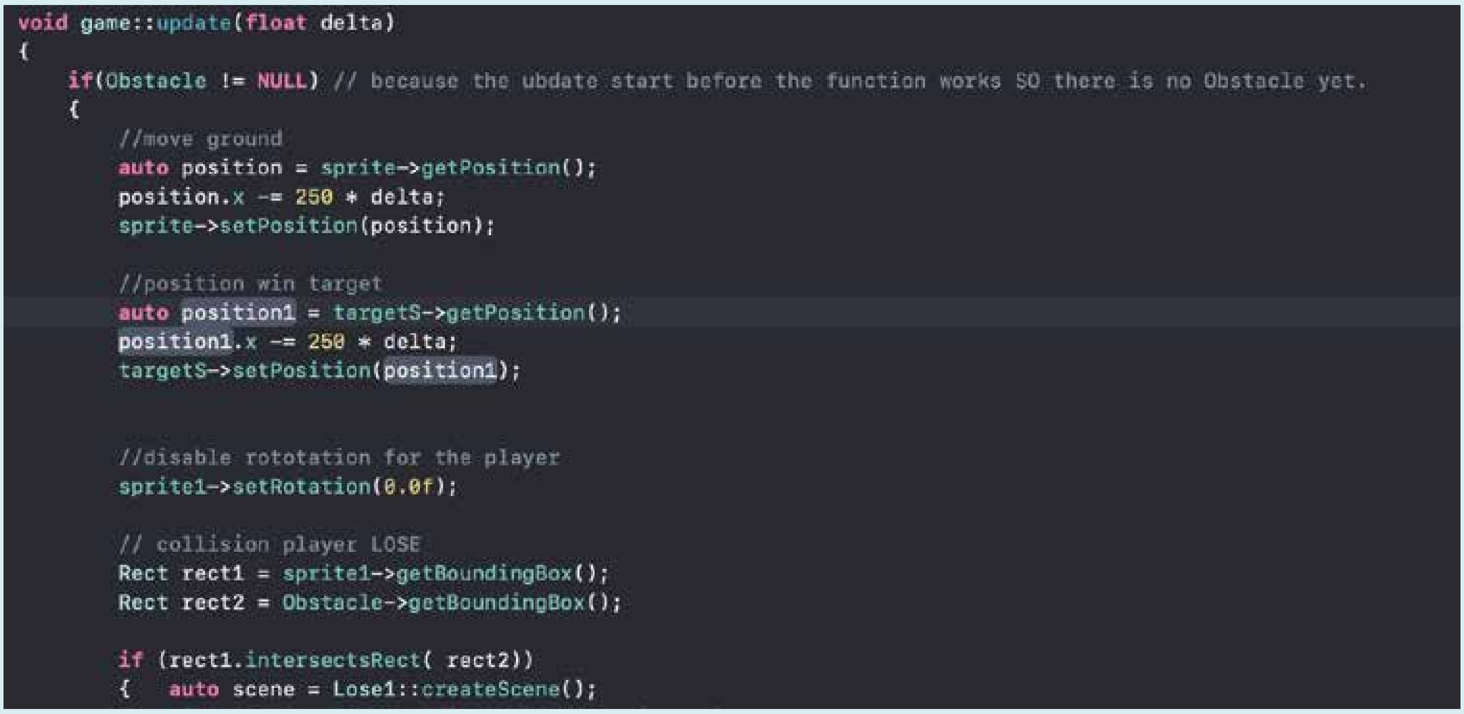
1- Declare the sprites.. (bool init)
2- Add physics to the sprites.. (bool init)
3- Ad keyboard listener. (bool init)
4- Function for pressing the button.
5- Moving ground code. (update)
6- Obstacles and target generator method.(Update)
7- If statement for collisions for win and lose.

_ This logic above will be the same 80% in the next levels_

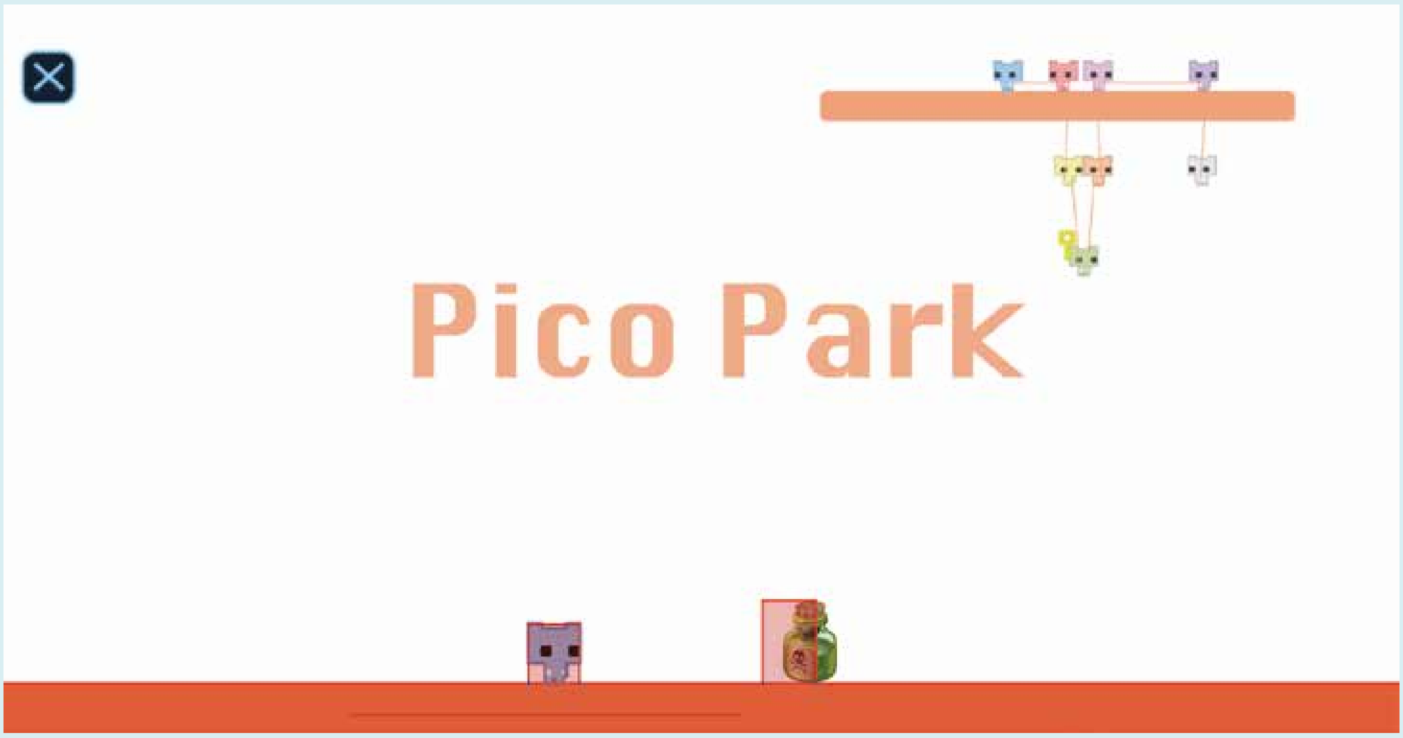
The problem that i had

The update function start before the init bool, wish was causing me a problem of NULL sprite .

In other word when the update run the Obstacle sprite is not generated yet.
That why i did a If statement to ignore if the value of Obstacle is NULL.



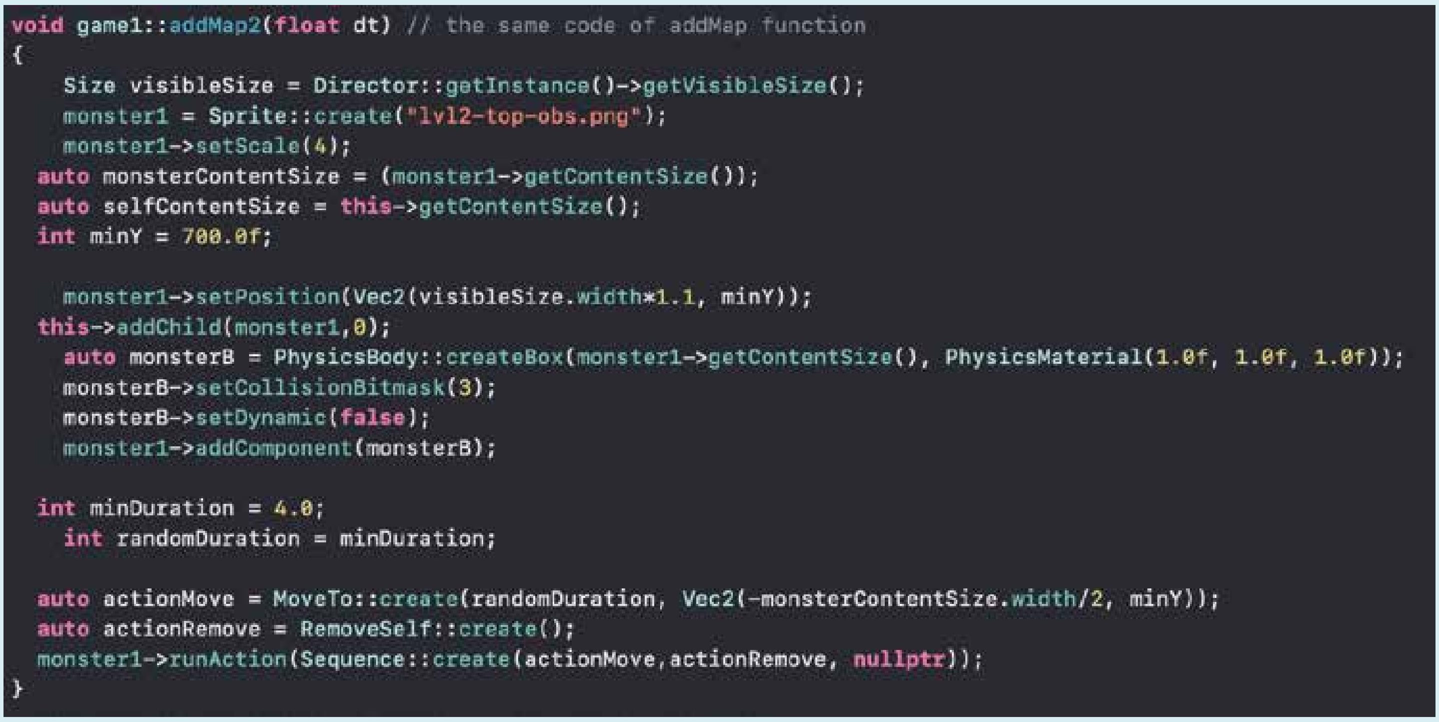
```
void game::update(float delta)
{
    if(Obstacle != NULL) // because the ubdate start before the function works SO there is no Obstacle yet.
    {
        //move ground
        auto position = sprite->getPosition();
        position.x -= 250 * delta;
        sprite->setPosition(position);

        //position win target
        auto position1 = targetS->getPosition();
        position1.x -= 250 * delta;
        targetS->setPosition(position1);


        //disable rototation for the player
        sprite1->setRotation(0.0f);

        // collision player LOSE
        Rect rect1 = sprite1->getBoundingBox();
        Rect rect2 = Obstacle->getBoundingBox();

        if (rect1.intersectsRect( rect2))
        {   auto scene = Lose1::createScene();
```



LEVEL 2

The Same logic but this time in order to make the game more challenging i will add another obstacle .
And the player will be a Plane this time.

I Just Duplicate the old function and change the Y poition.
Y = 700.0f in my case:



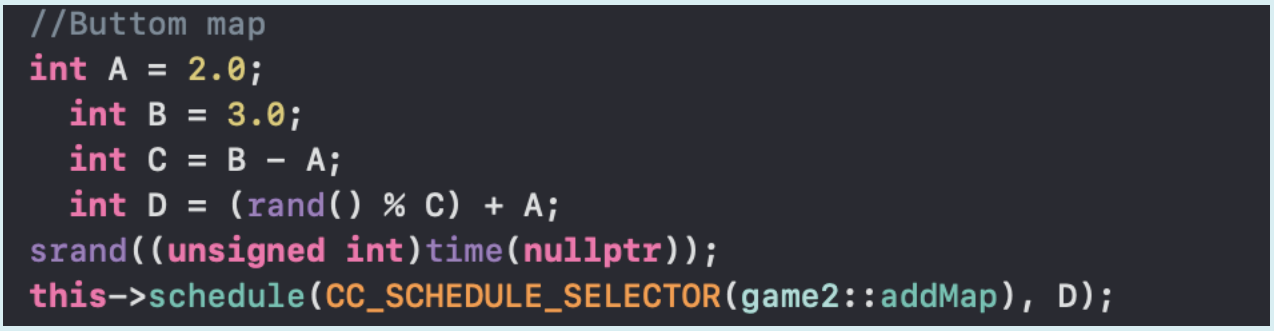
```
void game1::addMap2(float dt) // the same code of addMap function
{
    Size visibleSize = Director::getInstance()->getVisibleSize();
    monster1 = Sprite::create("lvl2-top-obs.png");
    monster1->setScale(4);
  auto monsterContentSize = (monster1->getContentSize());
  auto selfContentSize = this->getContentSize();
  int minY = 700.0f;

    monster1->setPosition(Vec2(visibleSize.width*1.1, minY));
  this->addChild(monster1,0);
    auto monsterB = PhysicsBody::createBox(monster1->getContentSize(), PhysicsMaterial(1.0f, 1.0f, 1.0f));
    monsterB->setCollisionBitmask(3);
    monsterB->setDynamic(false);
    monster1->addComponent(monsterB);

  int minDuration = 4.0;
    int randomDuration = minDuration;

  auto actionMove = MoveTo::create(randomDuration, Vec2(-monsterContentSize.width/2, minY));
  auto actionRemove = RemoveSelf::create();
  monster1->runAction(Sequence::create(actionMove,actionRemove, nullptr));
}
```

I made some changes to the gravity to make the plane fly more realistic .

LEVEL 3

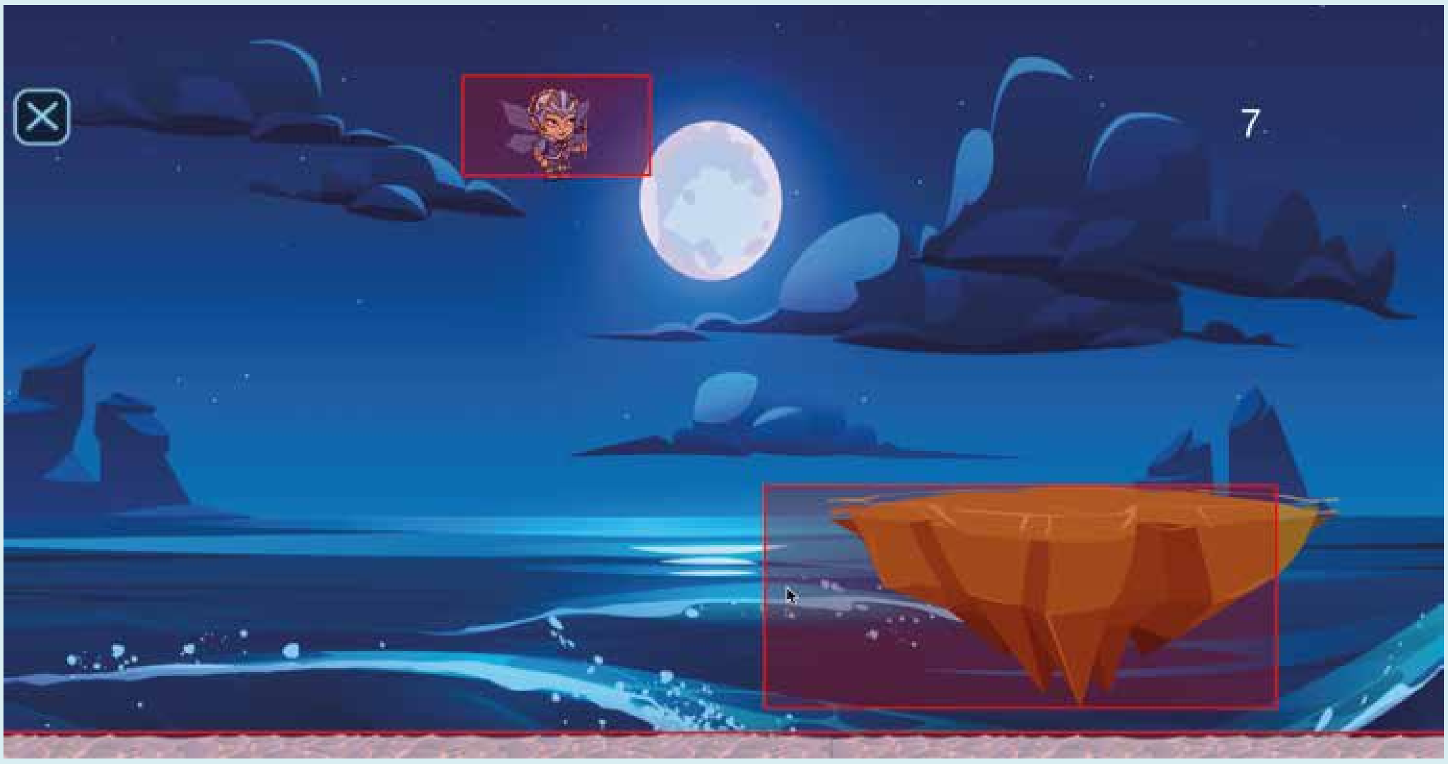
The same logic of level one again but this time the player needs to keep juming from land to land in order to win the level with top score

I made this lines to make the distance between the obstacles random.



```
//Buttom map
int A = 2.0;
  int B = 3.0;
  int C = B - A;
  int D = (rand() % C) + A;
srand((unsigned int)time(nullptr));
this->schedule(CC_SCHEDULE_SELECTOR(game2::addMap), D);
```

And i added the score the same i showed you but now we increase the score continully while the player is touching the ground.


7

LEVEL 4

This time the logic will change, I am going to use the rolling effect i showed before + the score increases every time the player touches the coin

Nothing special More than that.