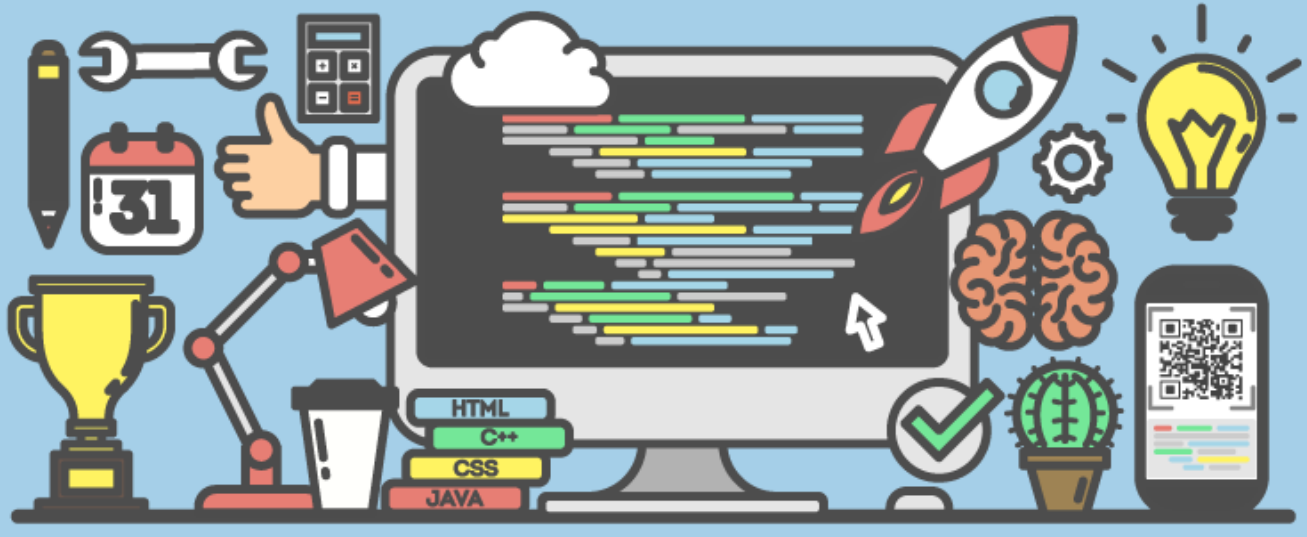
A TopDown Epic Adventure (with a Z)



Gaëtan Blaise-Cazalet - Head of programming section

The Sprite Class

The Sprite class

- In the previous games, we used always the same pattern for our game object classes :

We had some position and transformation variable : x, y, rotation, ox, oy...

We had some functions : Load, Update, Draw

- This is called boilerplate code. Code we always have to recreate.

- We would like to encapsulate this code into a class we will reuse through our next games. This class will be called the Sprite class.

The Sprite class

```
class Sprite
{
    public float X { get; set; }
    public float Y { get; set; }
    public float Ox { get; set; }
    public float Oy { get; set; }
    public float Rotation { get; set; }
    public bool Visible { get; set; }
    public Color Color { get; set; }

    Texture2D image;
    string path;

    public Vector2 Position
    {
        get
        {
            return new Vector2(X, Y);
        }
        set
        {
            X = value.X;
            Y = value.Y;
        }
    }

    public Rectangle Rect
    {
        get
        {
            return new Rectangle((int)(X + Ox), (int)(Y + Oy), image.Width, image.Height);
        }
    }

    public byte Opacity
    {
        get
        {
            return Color.A;
        }
        set
        {
            Color = new Color(Color.R, Color.G, Color.B, value);
        }
    }
...
```

The Sprite class

```
...
    public Sprite(int x, int y, string path)
    {
        X = x;
        Y = y;
        this.path = path;
        Color = Color.White;
    }

    public virtual void Load(ContentManager content)
    {
        image = content.Load<Texture2D>(path);
    }

    public void Draw(GameTime gameTime, SpriteBatch spriteBatch)
    {
        if (Visible)
        {
            Rectangle rect = new Rectangle(0, 0, image.Width, image.Height);
            spriteBatch.Draw(image, rect, null, Color, Rotation, new Vector2(Ox, Oy), SpriteEffects.None, 0);
        }
    }
}
```

The game objects classes

- From our sprite class, we can create different game object classes. They will inherit from Sprite.
- In our current game, we will want a Hero, a Projectile and an Enemy. They will all inherit from the Sprite class
- So create three Hero, Projectile and Enemy classes.

```
class Hero : Sprite
{
    public Hero(int x, int y, string path) : base(x, y, path)
    {

    }
}
```

```
class Projectile : Sprite
{
    public Projectile(int x, int y, string path) : base(x, y, path)
    {

    }
}
```

```
class Enemy : Sprite
{
    public Enemy(int x, int y, string path) : base(x, y, path)
    {

    }
}
```

Hero

Hero move

Make the Hero move up/down/left/right when pressing ZQSD.

We want a smooth move, like in the lander lesson.

Hero move : code

```
Vector2 speed;
const float ACCELERATION = 500;
const float DECELLERATION = 0.95f;
const float MAX_SPEED = 400;

public Hero(int x, int y, string path) : base(x, y, path)
{
    Visible = true;
}

public void Update(GameTime gameTime)
{
    float dt = (float)gameTime.ElapsedGameTime.TotalSeconds;
    KeyboardState ks = Keyboard.GetState();
    if (ks.IsKeyDown(Keys.Down))
    {
        speed.Y += ACCELERATION * dt;
    }
    if (ks.IsKeyDown(Keys.Up))
    {
        speed.Y -= ACCELERATION * dt;
    }
    if (ks.IsKeyDown(Keys.Right))
    {
        speed.X += ACCELERATION * dt;
    }
    if (ks.IsKeyDown(Keys.Left))
    {
        speed.X -= ACCELERATION * dt;
    }

    speed *= DECELLERATION;


    if (speed.X > 0) speed.X = Math.Min(MAX_SPEED, speed.X);
    if (speed.X < 0) speed.X = Math.Max(-MAX_SPEED, speed.X);
    if (speed.Y > 0) speed.Y = Math.Min(MAX_SPEED, speed.Y);
    if (speed.Y < 0) speed.Y = Math.Max(-MAX_SPEED, speed.Y);

    Position += speed * dt;
}
```

Hero rotation

Rotate the hero the right direction. You have two choices :

- Use the Rotation member of the Sprite class
- Create 4 textures in the Hero and display one in function of the direction

Projectiles

Projectile pop

Create a projectile class

Let's make a projectile appear on the right of the player

```
public Projectile(Hero hero) : base("projectile_horizontal")
{
        X = hero.X + 64;
        Y = hero.Y;
}
```

We need to change the Sprite class and add a constructor in it :

```
public Sprite(string path)
{
    X = 0;
    Y = 0;
    this.path = path;
    Color = Color.White;
}
```

And then add the Projectile in the Game1 class. The projectile will be created when we press Space (in the Update function) :

```
    Projectile projectile;
...
    protected override void Update(GameTime gameTime)
    {
        ...
        KeyboardState ks = Keyboard.GetState();
        if(ks.IsKeyDown(Keys.Space))
        {
            projectile = new Projectile(link);
            projectile.Load(Content);
            projectile.Visible = true;
            cooldownCounter = 0;
        }
        ...
    }
...
     protected override void Draw(GameTime gameTime)
    {
        ...
            projectile.Draw(gameTime, spriteBatch);
        ...
    }
```

Projectile List

The problem is we have only one projectile. We can have multiple projectile using a List<Projectile> instead of a single projectile variable.

```
List<Projectile> projectiles = new List<Projectile>();
```

```
protected override void Update(GameTime gameTime)
{
    ...
    KeyboardState ks = Keyboard.GetState();
    if(ks.IsKeyDown(Keys.Space))
    {
        Projectile projectile = new Projectile(link);
        projectile.Load(Content);
        projectile.Visible = true;
        projectiles.Add(projectile);
    }
    ...
}
```

```
protected override void Draw(GameTime gameTime)
{
    ...
    foreach (Projectile p in projectiles)
    {
        p.Draw(gameTime, spriteBatch);
    }
    ...
}
```

Projectile Cooldown

Our projectile generation is now continue. We have to add a cooldown to avoid that.

```
float cooldownCounter = 0;
const float COOLDOWN = 0.1f;
```

```
protected override void Update(GameTime gameTime)
{
    ...
    float dt = (float)gameTime.ElapsedGameTime.TotalSeconds;
    cooldownCounter += dt;

    ...
    if(ks.IsKeyDown(Keys.Space) && cooldownCounter > COOLDOWN)
    {
        ...
        cooldownCounter = 0;
    }
    ...
}
```

Projectile Direction : Hero

Now we want our projectile to go to the right when the player is orientated to the right, and to the left when the player is orientated to the left.

First we have to add a Direction property in the Hero class and to set the direction when we move.

```
public int Direction { get; set; }
```

```
public void Update(GameTime gameTime)
{
    ...
    if (ks.IsKeyDown(Keys.Down))
    {
        Direction = 2;
        speed.Y += ACCELERATION * dt;
    }
    if (ks.IsKeyDown(Keys.Up))
    {
        Direction = 8;
        speed.Y -= ACCELERATION * dt;
    }
    if (ks.IsKeyDown(Keys.Right))
    {
        Direction = 6;
        speed.X += ACCELERATION * dt;
    }
    if (ks.IsKeyDown(Keys.Left))
    {
        Direction = 4;
        speed.X -= ACCELERATION * dt;
    }
    ...
}
```

Projectile Direction

Then, in function of the hero's direction, we set the projectile when it is constructed.

We also set the move in function of the direction in the Update method.

```
int direction;
```

```
public Projectile(Hero hero) : base("projectile_horizontal")
{
    direction = hero.Direction;
    if (direction == 6)
    {
        X = hero.X + 64;
        Y = hero.Y;
    }
    else if (direction == 4)
    {
        X = hero.X;
        Y = hero.Y;
    }

}
```

```
public void Update(GameTime gameTime)
{
    if(direction == 6)
    {
        X = X + 5;
    }
    if(direction == 4)
    {
        X = X - 5;
    }
}
```

Do the same for the up and the down direction.

Tileset