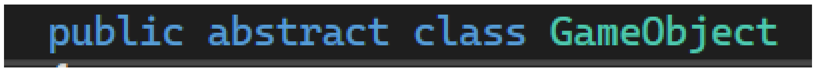
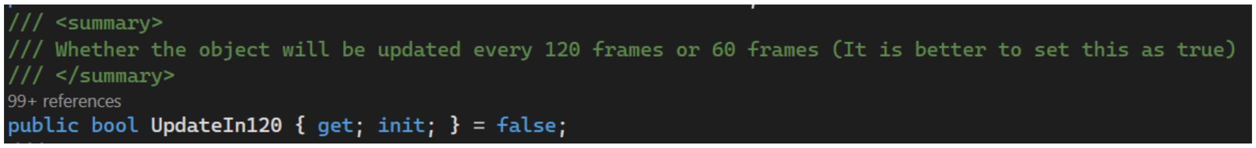
GameObject

Now that you have understood the basics of time calculation in this engine, we will briefly talk about how all entities function in the engine.
The fundamental class for all objects in the engine is GameObject



```
public abstract class GameObject
```

There are only a few variables you need to care about, one of them is UpdateIn120.

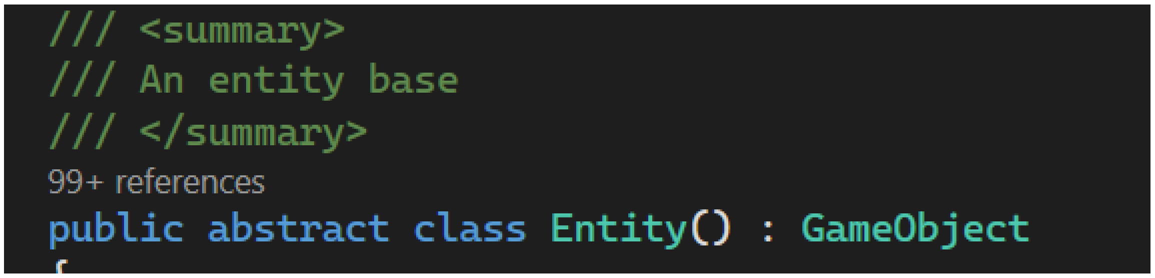


```
/// <summary>
/// Whether the object will be updated every 120 frames or 60 frames (It is better to set this as true)
/// </summary>
99+ references
public bool UpdateIn120 { get; init; } = false;
```

It controls the FPS of the object, it is very important so keep that in mind during coding a custom entity.
Another one is CurrentScene, it returns the current scene the game is in, CurrentFightingScene returns the current SongFightingScene, which you will be using a lot if the object is in a chart.

Entity

An entity is a derived class from GameObject, it is more complex



```
/// <summary>
/// An entity base
/// </summary>
99+ references
public abstract class Entity() : GameObject
```

You can use CreateEntity or InstanceCreate to create an Entity. (They are the same function)
You can create your own entities to achieve various effects.

Here are a list of variables you should keep track of:

AngleMode

Whether the entity uses Radians (true) or Degrees (false) as their angle

Centre

The centre of the entity

CollidingBox

The colliding box of the entity

controlLayer

The controlling surface of the entity (Default as Surface.Normal)
(Read about Surfaces in the chapter)

Depth

The depth of the entity (The higher the value is, the less shallow it is)

DrawOptimize

Whether the entity will be drawn regardless whether it is inside of the current view (false -> Drawn regardless, true -> Check if inside screen)

Image

The image of the entity to draw

ImageCentre

The centre of the image

Rotation

The rotation of the entity

Scale

The scale of the entity

SpriteBatch

The sprite batch that is used for drawing

Visible

Whether the entity is visible to the player

There are also some functions for the Entity class.

void FormalDraw(...)

This function draws a sprite.
There are a lot of overloads for this function, please check the documentation inside the engine for each individual use.

ShinyEffect CreateShinyEffect(Color? color = null, Texture2D image = null)

Creates an fade out effect of this entity
Color: The color of the effect
Image: The sprite used for the effect

void CreateRetentionEffect(float time, Color? color = null) => GameStates.InstanceCreate(new RetentionEffect(this, time, color ?? Color.White))

Creates a drag effect of this entity
Time: The duration of the drag
Color: The color of the drag

GravityEntity

This is a derived class from Entity, it contains three variables
Float Gravity: The magnitude of the gravity
Float GravityDirection: The direction of gravity
Vector2 Speed: The speed of the entity (The gravity will apply on this value)

Interface

Before we move into the next entity class, we need to talk about three interfaces that are crucial to understand.

ICustomMotion

This interface allows the entity to have a customized position route and customized rotation route. You can use Entities.SimplifiedEasing functions for it too.
Here are the variables for it

Func<ICustomMotion, Vector2> PositionRoute

The position function of the entity, use Motions.PositionRoute or SimplifiedEasing

float[] PositionRouteParam

The parameters for the position route

Vector2 CentrePosition

The centre position of the entity

Func<ICustomMotion, float> RotationRoute

The rotation function of the entity, use Motions.RotationRoute or SimplifiedEasing

float[] RotationRouteParam

The parameters for the rotation route

float Rotation

The rotation of the entity

float AppearTime

The frames elapsed after being created

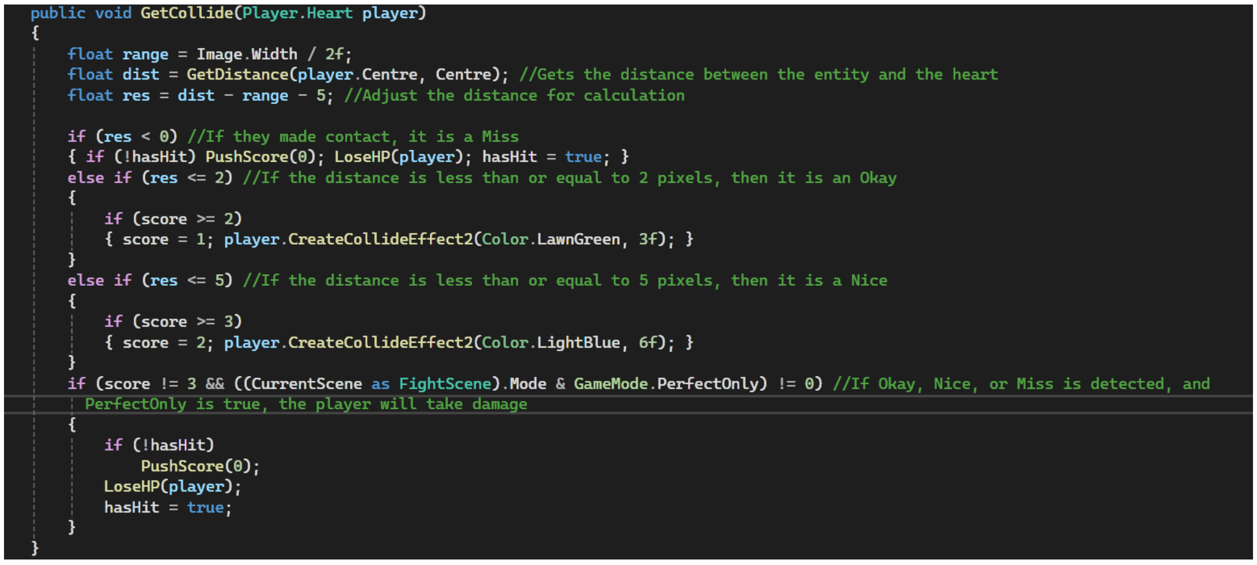
Motions.PositionRoute and Motions.RotationRoute may seem confusing, however there are already comments in the engine, so it should be easier to understand them.

ICollideAble

This interface indicates that this instance can collide with the player, it contains only 1 function.

GetCollide(Heart player)

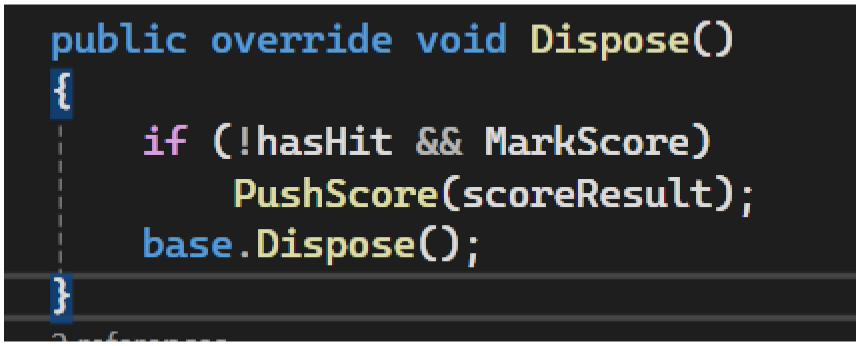
The function to check collision with the player
Player: The heart to check
A GetCollide() function should contain something like this



```
public void GetCollide(Player.Heart player)
{
    float range = Image.Width / 2f;
    float dist = GetDistance(player.Centre, Centre); //Gets the distance between the entity and the heart
    float res = dist - range - 5; //Adjust the distance for calculation

    if (res < 0) //If they made contact, it is a Miss
    { if (!hasHit) PushScore(0); LoseHP(player); hasHit = true; }
    else if (res <= 2) //If the distance is less than or equal to 2 pixels, then it is an Okay
    {
        if (score >= 2)
        { score = 1; player.CreateCollideEffect2(Color.LawnGreen, 3f); }
    }
    else if (res <= 5) //If the distance is less than or equal to 5 pixels, then it is a Nice
    {
        if (score >= 3)
        { score = 2; player.CreateCollideEffect2(Color.LightBlue, 6f); }
    }
    if (score != 3 && ((CurrentScene as FightScene).Mode & GameMode.PerfectOnly) != 0) //If Okay, Nice, or Miss is detected, and
      PerfectOnly is true, the player will take damage
    {
        if (!hasHit)
            PushScore(0);
        LoseHP(player);
        hasHit = true;
    }
}
```

And the Dispose() function should look something like this



```
public override void Dispose()
{
    if (!hasHit && MarkScore)
        PushScore(scoreResult);
    base.Dispose();
}
```

Barrage

Barrage is a derived class from Entity, and also adapts ICustomMotion and ICollideAble, so you can access all variables from these three. However, it has its own variables too.

Bool AutoDispose

Whether the barrage will automatically dispose itself when offscreen

bool MarkScore

Whether the barrage counts towards the score (Note that if the player gets a miss on this barrage, it will still count towards the final score)

int ColorType

The color type of the barrage (Can be user defined, but usually 0-> White, 1 -> Aqua, etc)

Color[] ColorTypes

The colors for each green soul shield
0-> Blue, 1 -> Red etc