# 🔥 Fireball 🔥

> 🚨 This feature is still **unpolished** but will be kept in the final version.

## Overview

> 💡 **FlappyFireball** is the class of the fireball enemies. Its script handle **movement** and **culling**.

The fireball script has **3 responsibilities**:

- **Culling** the object when **out of bound**.
- **Setting up** particles and properties of the object.
- **Moving** the object according to **random movement types**.

A script having multiple responsibilities at this point is *fine*, but it might need to be changed later.

## Readying the Fireball

> 💡 **Readying** is the step that executes **when all the node's children are present in scene.**

### Variable Initialisation

These **variables** using `@onready` are **defined only when all the children are in the scene.**
Then **using the `$` symbol** before a children's name (or path) will **store them into the variable.**

```
@onready var particles = $root_particle
@onready var hitbox = $hitbox
@onready var face = $face
```

### _ready()

This is the function called when the object is **ready**.
We call **three functions** that will be explained below.

- generate_properties()
- propagate_scale()
- setup_particles()

**Also** in this function there is a condition **setting the y velocity** to -60 if the object is a **free fall movement type**.

This is **bad** and will be **changed.**

```
func _ready():
	generate_properties()
	propagate_scale()
	setup_particles()

	# TODO: NOT HARDCODE THIS AND DO BETTER
	if movement == "fall_movement":
		velocity.y = -60
```

# generate_properties()

> 🚨 This function is **heavily hardcoded** and needs a **refactor**.

The below code **generates random values** for each property that **might be useful** for the fireball, such as:

- **movement**: the movement type of the fireball.
- **phase**: the offset of the movement (in case of **cosine wave**).
- **puls**: the higher it is, the higher the frequency of the **cosine wave** will be.
- **amp**: the amplitude of movement such as **cosine and triangular wave**.
- **wave_time**: the period of time it takes for a **triangular wave** to do one iteration.
- **scale_**: the scale of the fireball, **impact its speed and particle number**

```
func generate_properties():
	"""
	Initialise the fireball's property randomly.
	"""
	movement = movement_types.pick_random()

	phase = randf_range(0,2)
	amp = randi_range(50,120)
	puls = randi_range(2,8)
	wave_time = randf_range(1.2,2)

	scale_ = randf_range(1,1.75)
```

# propagate_scale()

> 🚨 This function **could need light refactor.** *(mostly as `speed` shouldn't be defined here)*

This function **changes according to scale**:

- **All particles scales** to be at minimum and maximum `scale_`.
- **Speed** which is inversely proportional to `scale_`.
- **Hitbox scale** which is equal to `scale_`.

```
func propagate_scale():
	"""
	Make the fireball faster the smaller it is.
	Also spread the scale to each particles.
	"""

	# Bigger the scale slower the fireball
	speed *= 1/scale_

	# Apply scale to every particles
	particles.scale_amount_min = scale_
	particles.scale_amount_max = scale_
	for particle in particles.get_children():
		particle.scale_amount_min = scale_
		particle.scale_amount_max = scale_
	hitbox.scale = Vector2(scale_, scale_)
```

# setup_particles()

> 🚨 Once again, **slightly hardcoded**.

**Set the amount of particles according to speed** using the formula:

$$amount = \frac{speed}{10}$$

```
func setup_particles():
	"""
	Increase the number of particles of the fireball according to its speed.
	"""
	particles.amount = int(speed / 10)
	for particle in particles.get_children():
		particle.amount = particles.amount
```

# Culling the fireball

## fireball_culling()

If the fireball is **too far** off the left or bottom edges, **delete it**.

```
func fireball_culling():
	"""
	When out of bound, remove the fireball from tree.
	"""
	if position.x >= CULL_DISTANCE and position.y <= -CULL_DISTANCE: return

	queue_free()
```

# Moving the fireball

## Movement system

Our current system work with an array of **function names** from which **one will be chosen** and attributed to `movement`.

Then it is called by using `call(movement)` in `_physics_process(delta)`.

The code **checks** each frame **if the fireball is out of bounds** via fireball_culling.

```
var movement_types = [
        "cos_movement",
        "linear_movement",
        "triangle_movement",
         "fall_movement"
        ]
var movement: String

func _physics_process(delta):
        """
        Move the fireball according to its random movement pattern.
        """
        time += delta
        velocity.x = -speed

        call(movement)

        move_and_slide()
        fireball_culling()
```

## Movement functions

Currently there are **four movement types**:

- **cos_movement()**: work in modular time (float) using `fmod(value, divisor)` .

$$\text{time} = \text{time} \mod 2\pi$$

- **linear_movement()**: which is just a `pass` since nothing needs to be done.
- **triangle_movement()**: work in modular time (float) using `fmod()`

$$\text{time} = \text{time} \mod \text{wavetime}$$

- **fall_movement()**: applies a vertical acceleration to fireball.

```
func cos_movement():
	"""
	Moves the fireball in the style of a cosine graph.
	"""
	time = fmod(time, PI*2)
	velocity.y = cos((time + phase) * puls) * amp

func linear_movement():
	pass

func triangle_movement():
	"""
	Moves the fireball in the style of a triangle wave graph.
	"""
	time = fmod(time, wave_time)
	velocity.y = sign(time - wave_time/2) * amp

func fall_movement():
	"""
	Moves the fireball like a free fall.
	"""
	# TODO: same, do not hardcode this and do better
	velocity.y += 0.7
```

# Styling the Fireball

Make it so the face and the particles are pointed in the direction of the velocity.

```
func _process(delta):
	"""
	Procedurally make the fireball look where it's going
	"""
	particles.direction = velocity.normalized()
	for particle in particles.get_children():
		particle.direction = particles.direction

	face.look_at(position + velocity)
```