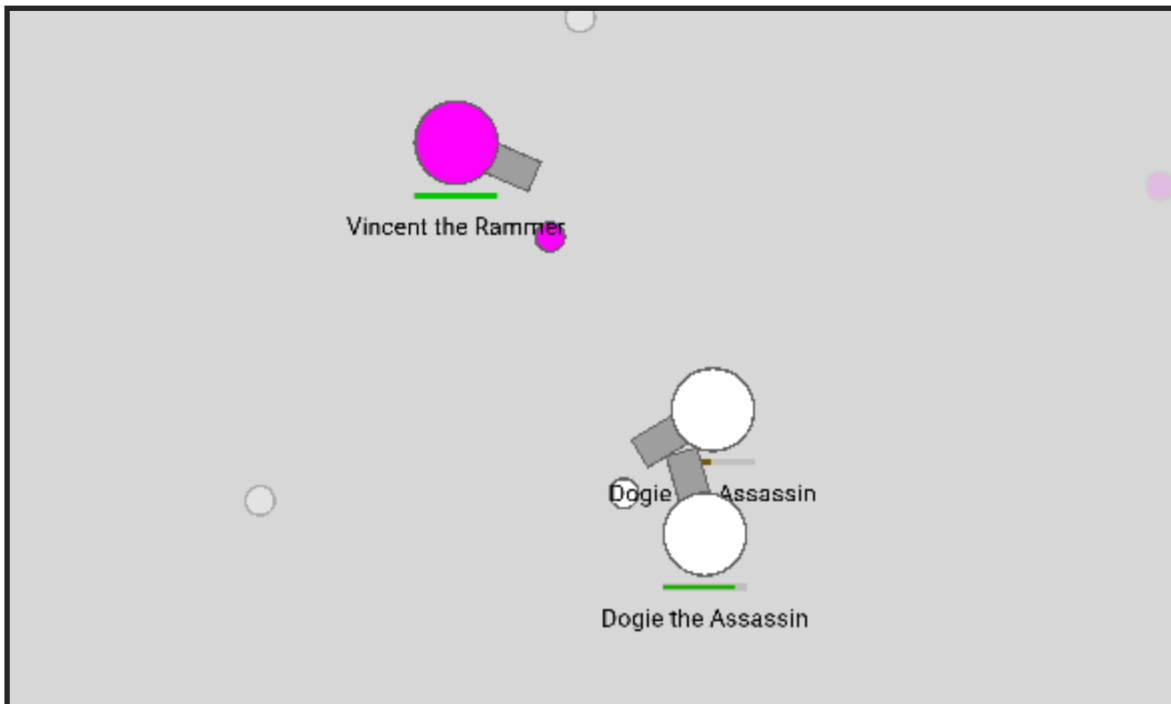


Automat**onk**

Player Manual

Introduction

Automatank is a **programming** game where you write a script that controls a tank to compete against other players. Each game has 4 players, and the last one standing wins. Spend **skill points** to upgrade your tank, manoeuvre around the map, and shoot bullets to be the victor!

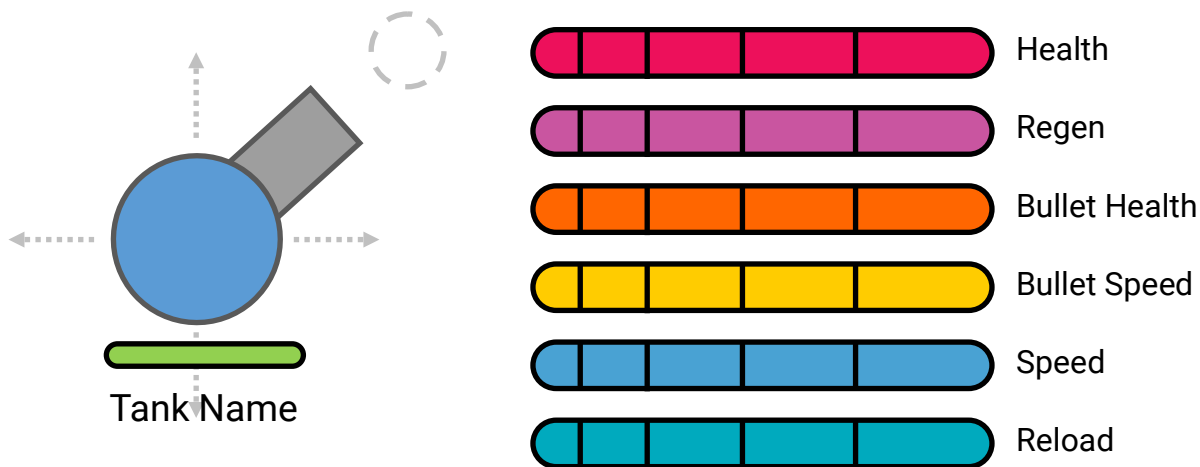


Game Mechanics

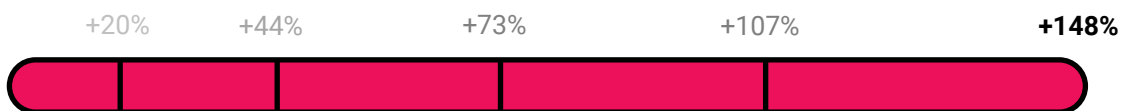
Rules

Each game of Automatank involves **four** tanks. The game starts after all tanks have spawned, and the last tank standing will be the winner.

Tanks

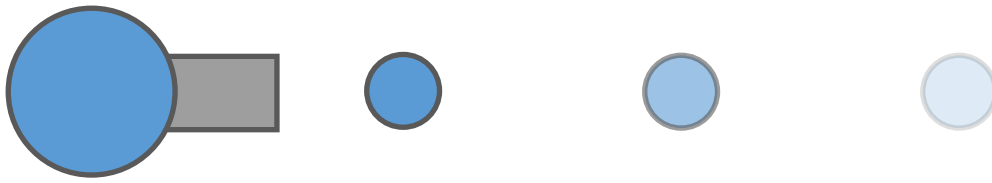


A tank is the unit you control in a game. It has a **radius** of **64** units. You may pick your own **colour** and your own **name**. You can freely **move** around and **shoot** bullets to defeat other tanks. Moving towards any direction **accelerates** the tank, and the tank will **slow down gradually** without any acceleration. A tank has a base **max health** of 100. When you collide with a **bullet** or **another tank**, you and the other body **both** rapidly lose **health** until one is destroyed. Usually, this means that you will lose a bit of health when you get hit by a bullet, but a tank with more health may try to physically **ram** the other tank to finish it off.

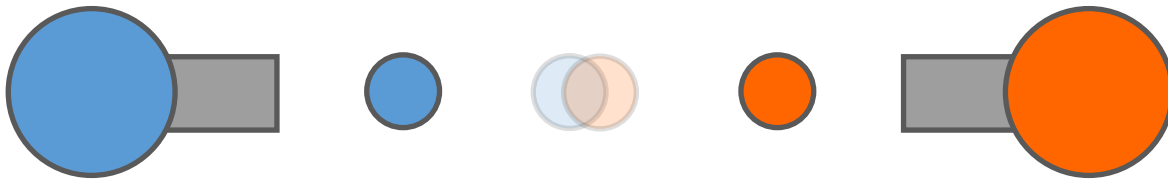


At the **start** of each game, you must spend **skill points** to **upgrade** some of your stats. You have **10 skill points** each game, and you may spend **up to 5** points on any **one given stat**. Upgrades on **one given stat** stacks **exponentially**, which means that the more points you spend on a single stat, the better the upgrades on that stat become. A fully upgraded stat is approximately **2.5x** as much as being un-upgraded.

Bullets

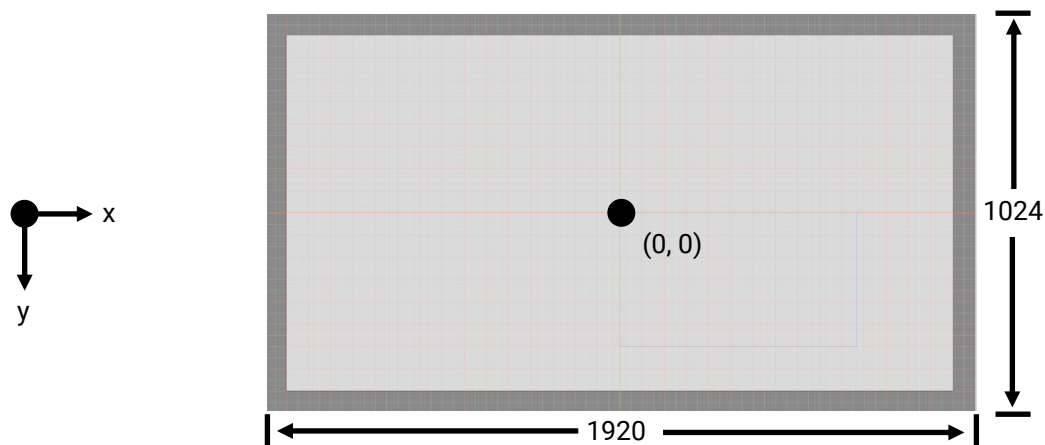


Tanks can fire bullets at any direction of their choice. They have a **radius** of **24** units, a base **health** of **25**, and a base **speed** of **750 units/second**. You can **upgrade** the bullet's **health** and **speed** to make them stronger, and upgrade your **reload** to shoot more bullets. Bullets travel at a **constant** speed and are not affected by friction, but **lose 10 health every second** until they disappear.



When a bullet collides with a **tank** or **another bullet**, both of them will rapidly lose health until one of them dies. This means that a bullet that hits another tank can deal damage up to its **health**, and that you can use your own bullets to block other bullets as a pre-emptive defence.

Arena

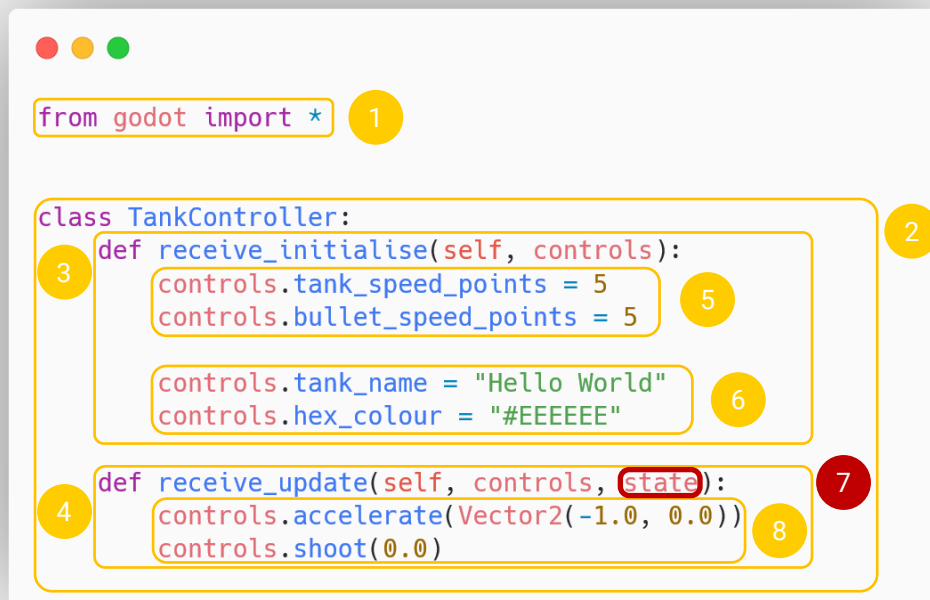


The arena of the game is a 1920x1024, fully open area with no obstacles, enclosed by walls. The centre of the arena is exactly **(0, 0)** within the game's coordinate system. The **x axis** points towards the **right**, the **y axis** points **down**.

Scripting

Tanks in the game are controlled by **Python** scripts that are written in a specific format. You can download a template for the script [here](#).

Sample Script



```
from godot import *

class TankController:
    def receive_initialise(self, controls):
        controls.tank_speed_points = 5
        controls.bullet_speed_points = 5

        controls.tank_name = "Hello World"
        controls.hex_colour = "#EEEEEE"

    def receive_update(self, controls, state):
        controls.accelerate(Vector2(-1.0, 0.0))
        controls.shoot(0.0)
```

- 1 Godot is the game engine that we use. Import it so we can use their functions.
- 2 Create a class called **TankController**. The game will create an instance of this class when it's loaded. **It has to be exactly TankController or the game will not load it.**
- 3 In the class, add a method called **receive_initialise(self, controls)**. This method will be called when the tank enters the game. You will spend your skill points for upgrades and customise your tank's appearance here.
- 4 Add method **receive_update(self, controls, state)**. This will be called a few times each second during the game. You will send instructions to the tank to move and shoot here.

5

Spend your skill points to upgrade your tank. **You have to spend all of your skill points in `receive_initialise`. Spending points elsewhere in the script has no effect.** Use `controls.*insert_skill_point_name* = *points here*` to spend your skill points.

The following is a list of skill point names:

Stat	Skill Point Name
Max Health	<code>tank_speed_points</code>
Regen	<code>regen_points</code>
Bullet Health	<code>bullet_health_points</code>
Bullet Speed	<code>bullet_speed_points</code>
Speed	<code>tank_speed_points</code>
Reload	<code>bullet_cooldown_points</code>

6

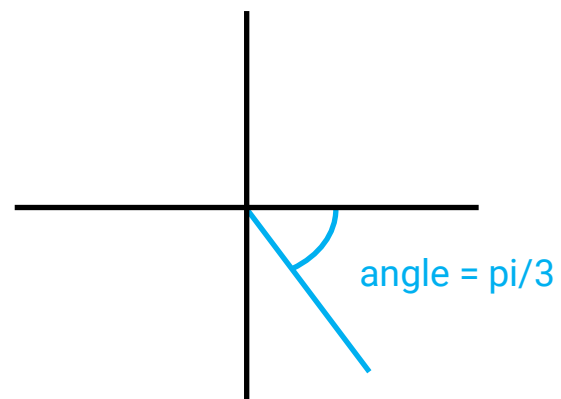
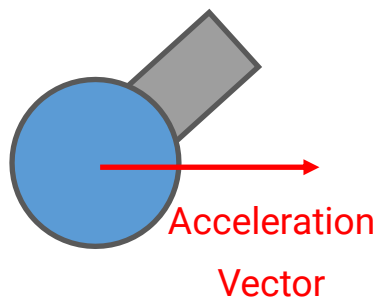
Customise your tank. Use `controls.tank_name = "name"` to set your tank's name, and `controls.hex_colour = "colour"` to set your tank's colour. The game uses Hex colour codes to encode colours, you can pick your colour and copy the code from [here](#). **You have to customise your tank here. Customising elsewhere in the script has no effect.**

7

The game state will be passed to you as a parameter. Learn more [here](#).

8

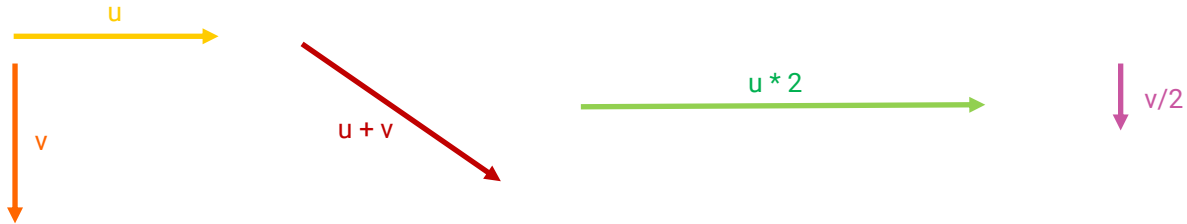
Control your tank. Use `controls.accelerate(*insert vector*)` to accelerate your tank in the given direction. The parameter for this is a **Vector**, learn more about working with vectors [here](#). Use `controls.shoot(*insert angle*)` to shoot a bullet in the given direction. **This angle is in radians, going clockwise from the x axis because the y axis points down.**



Working with Vectors

```
vector = Vector2(x, y)
```

Many of the objects and concepts in the game are represented by **vectors**, including any **position**, **velocity**, and **acceleration**. Vectors are very powerful tools that can make calculations dramatically easier. To create a vector, use the constructor **Vector2(x, y)** and provide it the x and y values of the vector you want to create.



You can add and subtract vectors, or multiply and divide them by constant values to obtain different vectors based on your existing vectors.

On top of this, there are a lot of methods built-in to the Godot vector that you can use to help you. Here is a list of the most commonly used ones:

Method	Description
<code>vector.angle()</code>	Returns the angle of this vector to the x-axis, in radians.
<code>v.angle_to(other)</code>	Returns the angle between this and another vector.
<code>v.distance_to(other)</code>	Returns the distance between the tips of the vectors.
<code>v.dot(other)</code>	Returns the <i>dot product</i> of the two vectors.
<code>v.length()</code>	Returns the length of the vector.
<code>v.normalized()</code>	Returns a vector of equal direction but length 1.
<code>v.rotated(angle)</code>	Returns the vector rotated by the given angle (radians).

GameState

A GameState is an object passed to you in the **receive_update** method of your TankController class. You can use this object to check your own state and the state of all tanks and bullets.

You can learn the following about yourself from the game state:

Field	Type	Description
state.health	number	Your current health.
state.max_health	number	Your maximum health.
state.position	Vector2	Your position.
state.velocity	Vector2	Your velocity.
state.bullet_cooldown	number	The number of seconds until you can shoot.

You can obtain information about the world from the game state too. Here are the things that you can access:

Field	Type	Description
state.tanks	list	List of all surviving tanks in the world.
state.bullets	list	List of all bullets in the world.

Keep in mind that **state.tanks** and **state.bullets** are **lists**. You can access individual tanks and bullets by indexing the lists (**state.bullets[0]**) or by using **for loops** (**for bullet in state.bullets**). The list of tanks and bullets *include yourself and your bullets*.

When you have access to an individual tank a bullet, you can access these fields to obtain information about them (here the tank or bullet in question is referred to as "body"):

Field	Type	Description
body.health	number	The body's current health.
body.friendly	boolean	True if the body cannot harm you (your bullet).
body.position	Vector2	The body's position.
body.velocity	Vector2	The body's velocity.

Playing

After writing down your tank script, please **save** it somewhere on the computer as a Python file. **If your script triggers any error, the error will show up on the bottom right corner of the screen and your tank will be immediately removed.**

Test Mode

You can enter **test mode** by clicking the test mode button at the main menu. Once you are in test mode, you can **drag your script file from your file explorer directly into the screen** to spawn your tank on the spot.

To test the abilities of your tank, you can spawn dummy tanks—tanks that do nothing—with the **Spawn Dummy Tank** button, or you can go in yourself with the **Spawn Player** button. Use WASD to move and click to shoot. The tank you control as a player has maximum upgrades (all stats doubled). You may think that this is redundant, but you will need this to win against your own scripts later!

If your script tank has gone out of control, or you would like to start a new round, you can use the **Kill All** button to clean up the battlefield.

Play Mode

You can play real games by clicking the play button at the main menu. Once you are in the game, you can **drag script files into the screen** to spawn tanks. Each game requires four players, and once four tanks are in position, the game will start.

Once a winner is determined, the game will show the winner's name. *The game does not pause after it's finished, so if you have spare time you can write up a victory dance to show off your power!*

Good luck coding!

Now that you've learned and mastered the game and scripting, go off and write your own AI and have some fun!