Creative Computing.

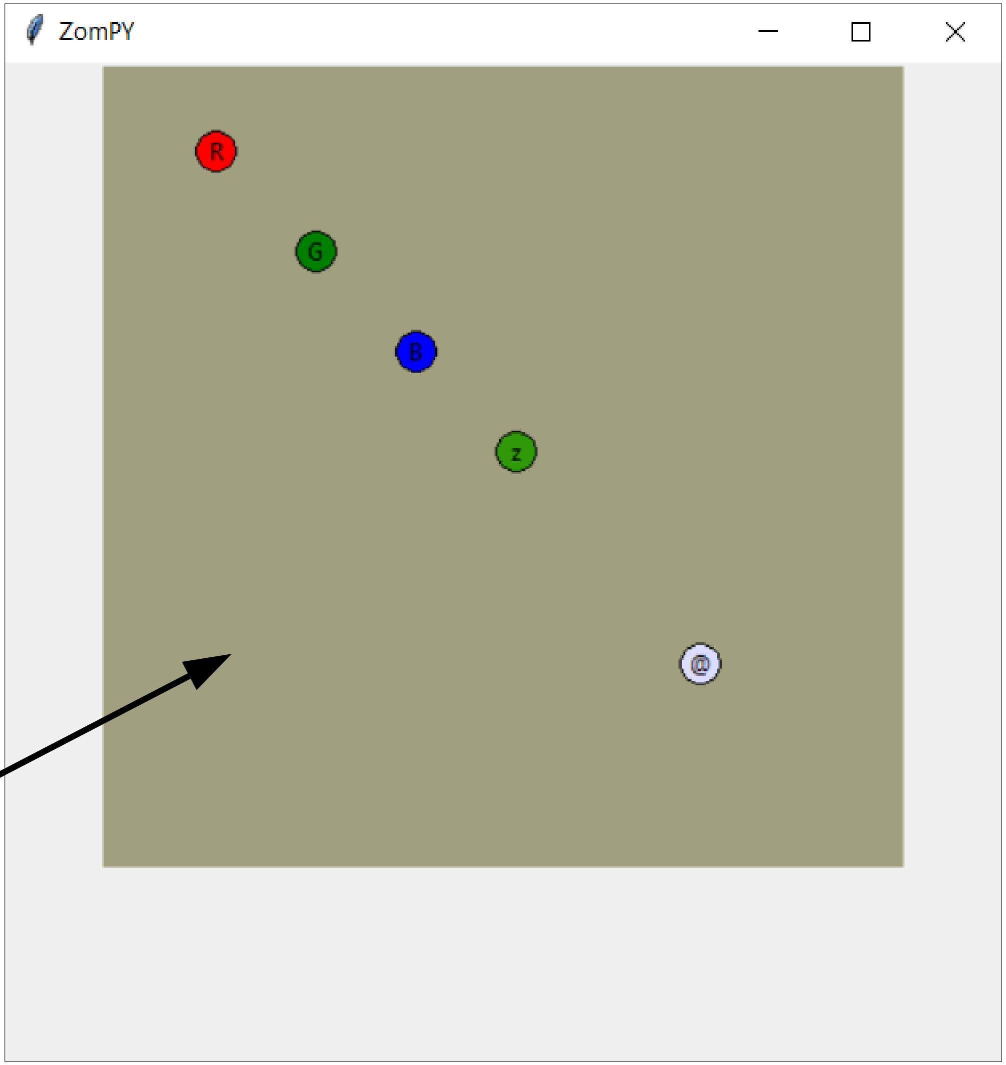
(doing something cool, with computers)

What are we doing here?

- We’re going to make a video-game!
- With guns…
- And Zombies…
- In Python…
- Using tkinter for graphics…
- And a custom game engine.

Introducing… ZomPY!

- We're going to make a video-game!
- With guns…
- And Zombies…
- In Python…
- Using tkinter for graphics…
- And a custom game engine.
- And it will look like this *(next-gen graphics I know!)*

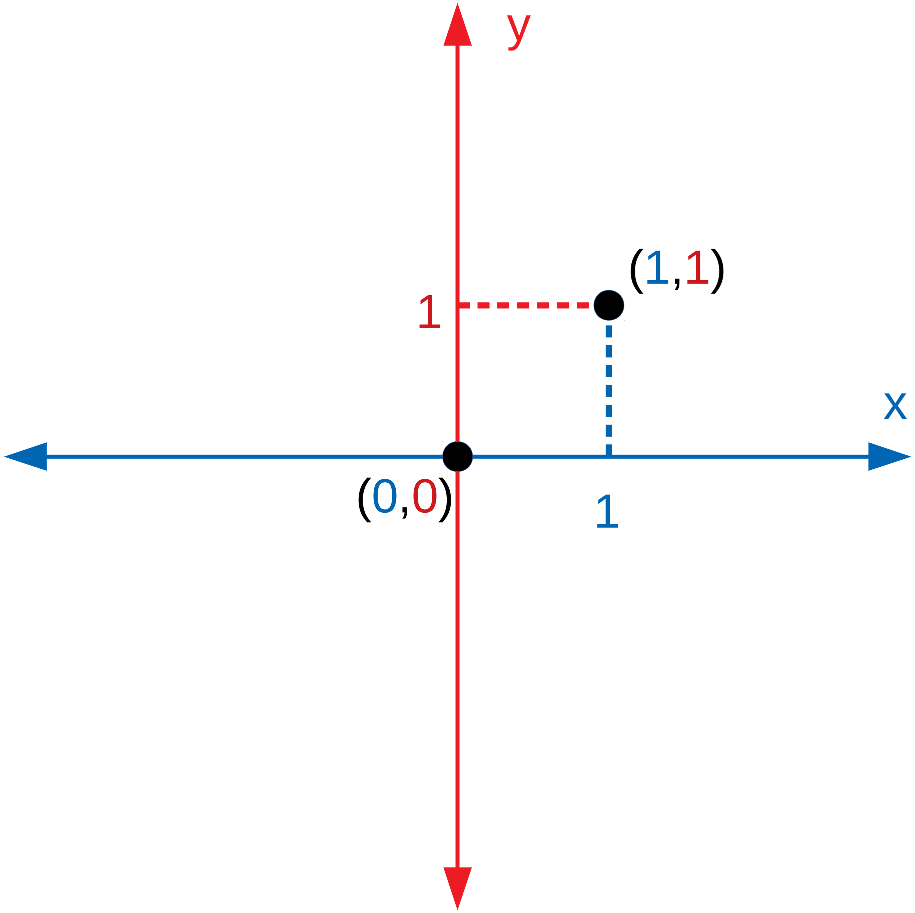


Game Engine Design

(to use the engine you must first understand it, or at least part of it!)

2 Dimensional (2D) Space...

- Our game world is a flat 2D space.
- 2D space means that every place/point in this world has 2 coordinates.
- Points have X and Y coordinates, specifying how far along each axis it is.
- Think of it like a graph in maths...
- The origin is at (0,0)
- Oh... And coordinates can be negative too!



A note about maths…

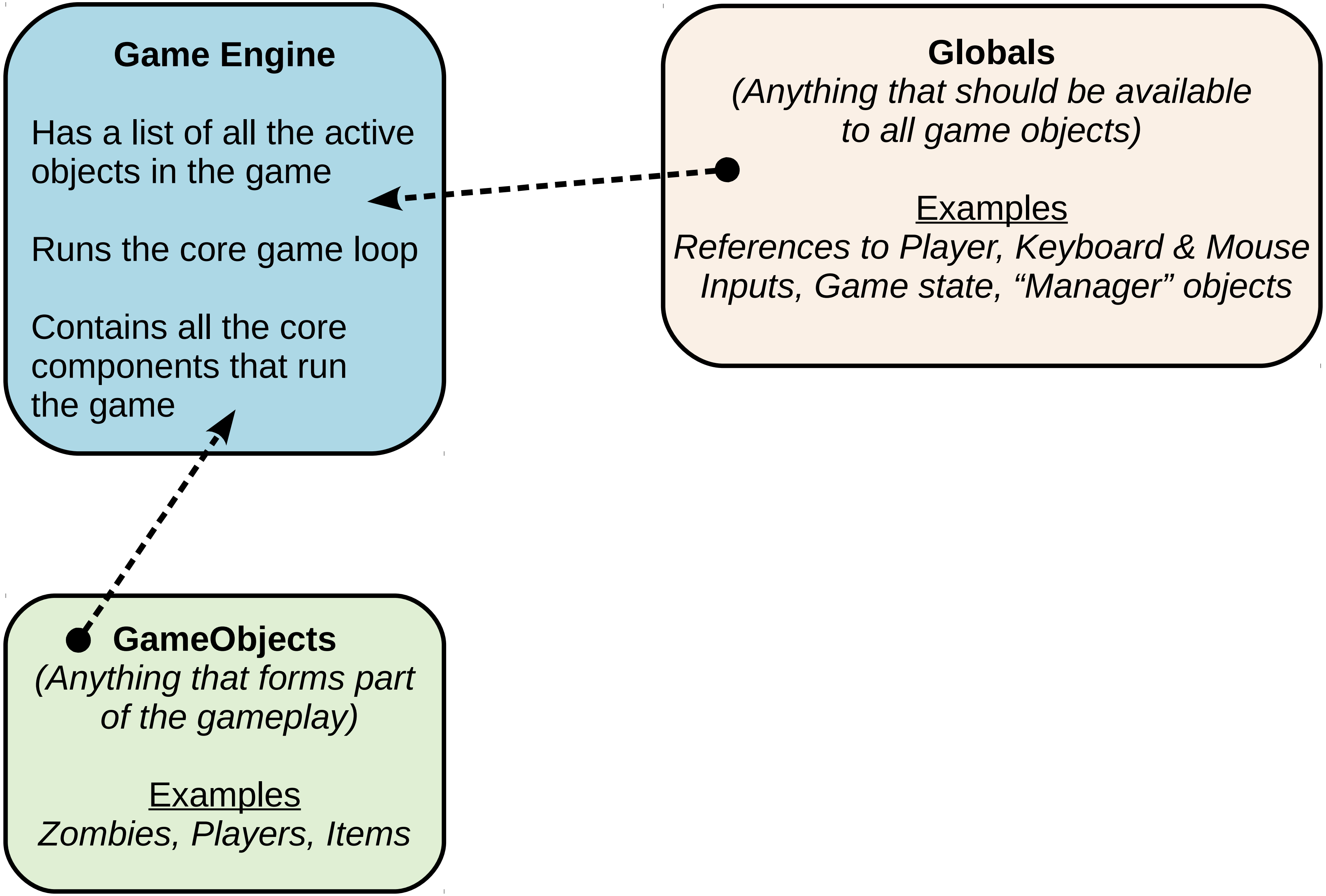
- There is a baseline of maths that is required, 2D coordinates mainly.
- But it can become as mathematically complex as you desire!
- *Warning: Some of the maths in this project (traceline in particular) is A-Level standard.*
- I am happy to explain all the maths to those who want to know.

- **Don’t worry if you don’t understand the maths! Any maths methods or functions we need I’ll provide for you!**

Your world is full of (stupid) objects.

- Introducing…. The ***GameObject***!
  - *As the name suggests, it's an object that exists in our game.*
- GameObjects are updated every game frame and (optionally) drawn into the game world.
- The core part of the programming resides around objects. But they won't do anything unless you tell them to!
- I've provided a sample player object and a zombie object for you to look at.

The Core Engine Design



Game Engine

Has a list of all the active objects in the game

Runs the core game loop

Contains all the core components that run the game

Globals

(Anything that should be available to all game objects)

Examples

References to Player, Keyboard & Mouse Inputs, Game state, “Manager” objects

GameObjects

(Anything that forms part of the gameplay)

Examples

Zombies, Players, Items

The Fun Part.

(actually coding stuff)

Getting Started

- I’ve hosted the basic code on my GitHub.

- Go to: *github.com/jamesadey/zompy*
- Download all the files
  - *There should be a button to download/clone*
- Open zompy_launcher.py in IDLE
  - Python version 3.x please!
- Run this file, and the game should start…

A Game Engine… and a Game!

- Core Game Engine
  - engine.py
  - mainwindow.py
  - gameobject.py
  - gameworld.py
  - example_graphics_object.py

- Game… ZomPY
  - Everything else!

...But why split it?

- Good practice to write decoupled code that can be adapted to many circumstances.
- I wanted the game engine to be suitable for any game type (not just zombies).
- So I can give you guys something to take away from this project to go and explore making your own games!

What happens when I run it?

- Launcher
  - Create engine → start_game()
- Engine → **make_globals()**
  - **Create and initialise your global data structure here**
- Engine → **setup_game()**
  - **Do all of your game setup here! Add objects, initialise managers, etc…**
- Engine → run_game()
  - Hook into tkinter to form a game loop

What to do first...

- The current game is a bit... Lacking?

- So 2 tasks to get you used to the engine!
- 1) Make the zombies follow the player
- 2) Give the zombies health so they can be shot

1) Make zombies follow the player

- A few key questions to ask ourselves…
- How do the zombies know where the player is?
- Once we know where the player is… How do we move towards them?

1) Make zombies follow the player

- A few key questions to ask ourselves…
- How do the zombies know where the player is?
  - How do zombies know about the player?
  - ***Use our Globals for storing the player!***
- Once we know where the player is… How do we move towards them?
  - We want our coordinates to be the same as the player’s coordinates…
  - ***Use if statements!***
  - ***If our player has a higher X coordinate than us, then we want to move in the positive X direction…***
    - Repeat for all 4 directions!

2) Give the zombies health so they can be shot.

- Again, another few key questions…

- How does the zombie know how much health it has?

- How do we know if we hit a zombie?

- How do we damage a zombie?

- How does a zombie know if it’s dead?
  - *On a philosophical note… Can the undead die? Hmm…*

2) Give the zombies health so they can be shot.

- Again, another few key questions…
- How does the zombie know how much health it has?
  - ***Use a variable to remember our health!***
- How do we know if we hit a zombie?
  - This is more tricky… How can we tell if an object is a zombie?
  - ***We can to use a python function to check if what we hit is an instance of a zombie!***
- How do we damage a zombie?
  - ***We need a method on the zombie that we can call (from the player) to tell it to take damage!***
- *How does a zombie know if it’s dead?*
  - ***When it’s health goes below zero, it is dead.***
  - ***When a zombie dies, remove the zombie from the game!***

More? More you say?
Fine, sink your teeth into this...

- More zombies, we need a horde.
  - 4 zombies isn't nearly enough, add some more!
  - *How many can the game handle before lagging?*
- Zombies just die and the game ends too fast
  - Make the zombies re-spawn when they die.
- Our player is invincible, and zombies don't hurt it.
  - Firstly, give the player some health.
  - Then make the zombies attack the player.