# Are zombies following the navigation grid?

- We didn't quite get the gridwalker zombie following the navgrid last week.
- Finish this off as a starter.
- If you need them, you can get the slides from last week here and work through adding them in:

**github.com/JamesAdey/zompy/blob/master/slides/cc_week4.pdf**

- Afterwards, download the slides for this week from here:

**github.com/JamesAdey/zompy/blob/master/slides/cc_week5.pdf**

# Creative Computing.
# Week 5

(cool stuff + computers = ?)

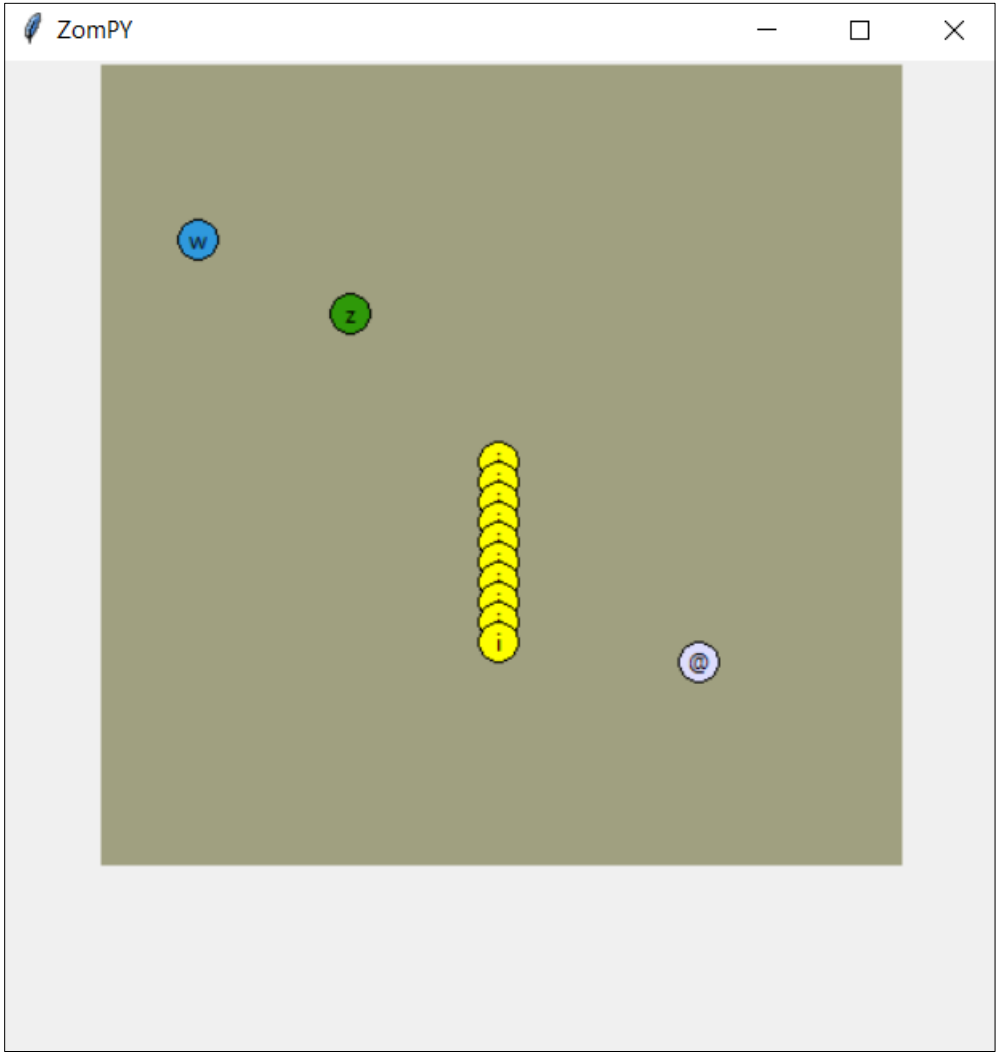
# If you don't have the project yet...

- I've hosted the basic code on my GitHub

- Go to: ***github.com/jamesadey/zompy/tree/week5***
  - *This is the project containing finished code from week 4, ready for week 5.*
- 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…

# Where next?

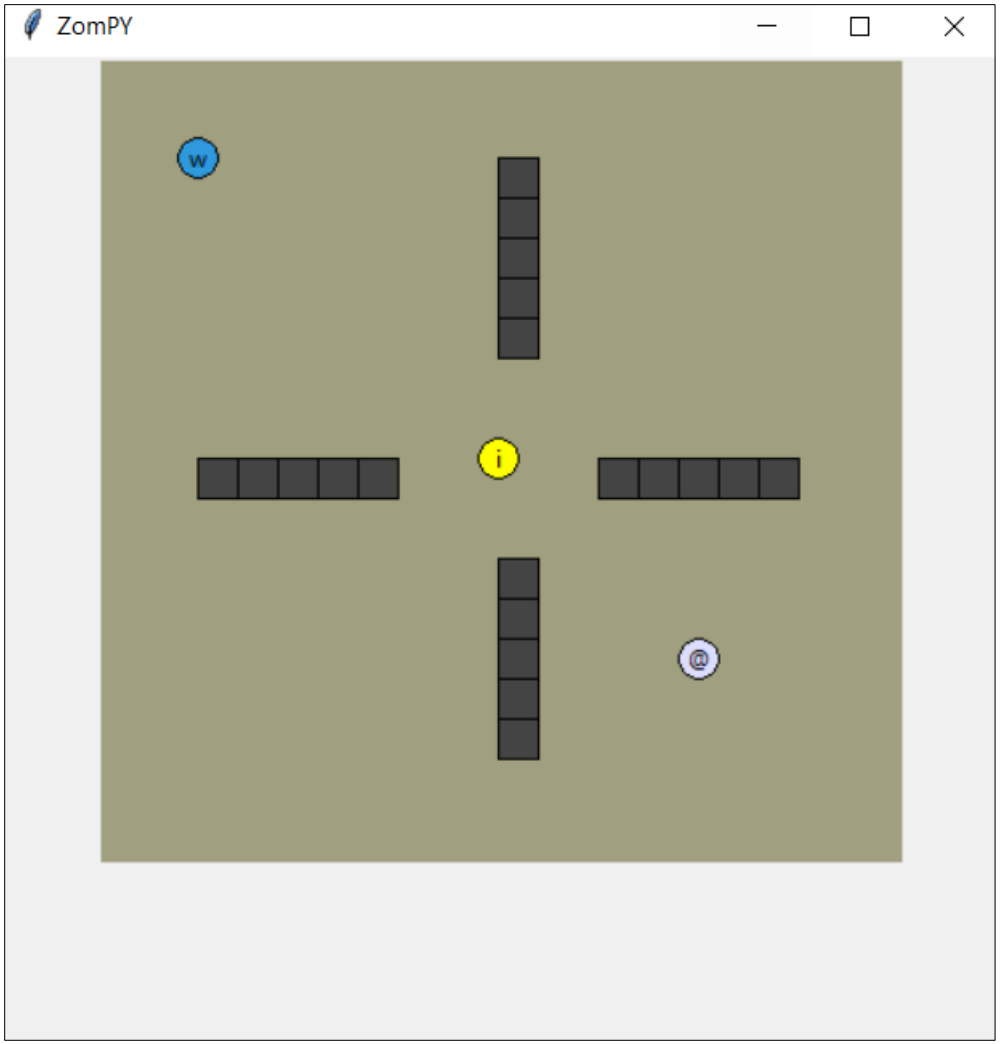
- Our zombies can move… intelligently!
- And they can be shot…
- We have spawners…
- And item pickups!



# Levels!

- Our zombies can move… intelligently!
- And they can be shot…
- We have spawners…
- And item pickups!

- Now it's time for some walls!



# Random maps.

(and bugfixes!)

# A more practical session...

- Less theory today, more practical programming! We'll split the lesson into two parts.
- Part 1: Map generation
  - Adding in the map generator, and moving game setup to a level based system.
- Part 2: Bugfixes!
  - Fixing a bug with the player radius being zero

# Map Generator

- This is a module, instead of a class.
- It contains helpful functions to use when creating a level.

- ***create_random_rocks (gameEngine,width,height,numwalls)***
  - This function places random rocks/walls around the box specified by width and height
- ***create_rock_wall(gameEngine,startX,startY,endX,endY)***
  - This function creates lines of rocks/walls from (startX,startY) to (endX,endY)

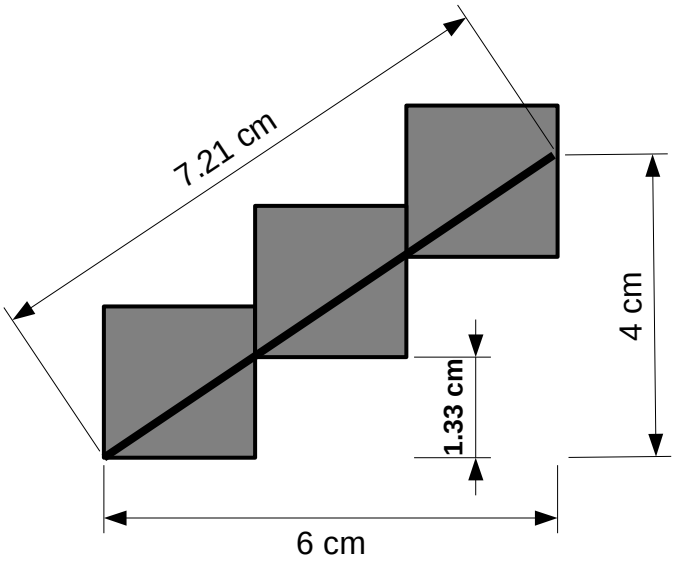
# Creating lines of walls.

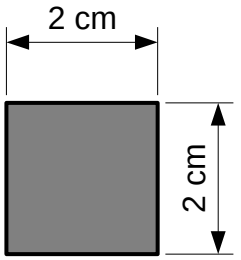
**Algorithm to create lines of walls**

**Inputs: *wallSize*, *startX*, *startY*, *endX*, *endY***

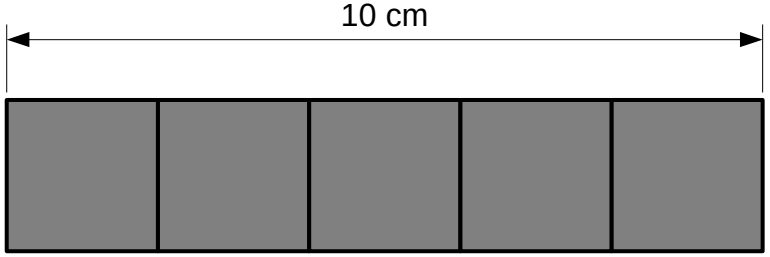
1) Work out how many walls we need
2) Work out the x and y step
3) For each wall (i):
    x = startX + halfSize + (stepX * i)
    y = startY + halfSize + (stepY * i)
    create a wall at (x,y)

Count = 7.21 / 2 = 3
Step X = 2
Step Y = 1.33




Count = 10 / 2 = 5
Step X = 2
Step Y = 0



# A M*odule* not a *Class*

- So far we've had 1 class per module, but for this code we are working with groups of functions instead of with concrete objects
- The Map Generator is not a class, but is instead a collection of functions that can be used to spawn walls across a map.
- Levels are just functions (methods) that create all the obstacles, spawners and things in each level, and add them to the game engine.
- These are more like configuration methods than actual objects in our game.
  - These might also add the player in too! Don't forget that!

# The Fun Part.

(actually coding stuff)

# Download these slides.

**github.com/JamesAdey/zompy/blob/master/slides/cc_week5.pdf**

# Integrating the Levels

- I've provided you with a basic map generation module, as well as 2 levels and a rock wall item.
- 4 more files to add to the project:

  ***github.com/JamesAdey/zompy/blob/week5/level_random.py***

  ***github.com/JamesAdey/zompy/blob/week5/level_cross.py***

  ***github.com/JamesAdey/zompy/blob/week5/mapgenerator.py***

  ***github.com/JamesAdey/zompy/blob/week5/rockwall.py***

- Then follow the instructions in the following slides.

# Edit: *zompy_engine.py*

- *Remember to add the import statements, only import the file as a module, instead of all the contents.*

```
from item_manager import *
from navgrid import *

# these are not a class currently
# but a suite of helper methods
import level_random
import level_cross


class ZompyGlobals(GameGlobals):
    zoms = 10
    bulletManager = None
```

- ***Inside the setup_game function:***
- *Create a variable to hold the current level name*
- *Then check the level name and call the correct generation function*

```
# Create a nav grid
ng = NavGrid(resolution=20)
super().add_game_object(ng)
gGlobals.navGrid = ng

levelName = "cross"

if(levelName == "random"):
    level_random.create_level(super(),gGlobals)
elif(levelName == "cross"):
    level_cross.create_level(super(),gGlobals)
else:
    print("ERROR Could not create level!")
```

# Create: *level_custom.py*

- *Following the template in the existing levels add any required import statements!*
- *Define a create_level function*
- ***Cut** and **paste** any blocks of code containing level setup information **from zompy_engine.py***
- *Change any references to **super()** into **gameEngine***
- *Don't forget to add a player to your custom level!*

**level_custom.py**

```
from player import *
from example_item import *
from zombie_gridwalker import *
from zombie import *
# this is not a class currently
# but a suite of helper methods
import mapgenerator

def create_level (gameEngine, gameGlobals):

    '''
    ------------------------------------
    Add Your Level Generation Code Here
    ------------------------------------
    '''
    # create a zombie spawner
    zs = ZombieSpawner(x=100,y=100)
    gameEngine.add_game_object(zs)

    # ALWAYS remember to create the player!
    pl = Player(x=300,y=300)
    gameEngine.add_game_object(pl)
    gameGlobals.player = pl
```

**zompy_engine.py**

```
gGlobals.itemManager = im
super().add_game_object(im)

# create a zombie spawner
zs = ZombieSpawner(x=100,y=100)
super().add_game_object(zs)

# create a nav grid
ng = NavGrid(resolution=20)
```

# Play it!

- Now is the time to test and see if your changes work!
  - You should notice that the walls block bullets and shooting (but not the lines drawn). I'll get a fix for this in the future.
  - Walls also don't block player movement, but you could make them slow the player down…

- **Can you play the two example levels?**
- **Can you modify the code to play your custom level?**

- Once you're done, continue on to the bugfixes slides.

# A bugfix!

- There is a bug with the player code, that means that it's collision radius is always zero, so collisions aren't being detected correctly.
  - It inherits the function from GameObject, which by default returns zero.
  - *From a code architecture perspective… Would it be better for GameObject to not define this function, and throw errors when running, maybe this bug would have been found sooner…*
- Adding a function definition for *get_collision_radius* should fix this.
- In **player.py**, add this function:

```
def get_collision_radius(self):
    return self.radius
```

- This should the bug, and the player should detect collisions with walls as soon as they touch.

# Free Play & Extensions.

- Now it's unguided work! But here's some suggestions:
- **Create your own levels!**
  - Keep editing level_custom.py or make an entirely new one!
  - Can you define your own random generation functions?
- **Add obstacles that damage zombies and players**
  - Try editing the *on_collision* function to send a take damage message when an obstacle collides with a player/zombie.

# Feedback

- Did you enjoy the more practical session?
- Are the code examples included on the slides helpful?

- We have 3 weeks left, and there's still more I want to cover.
  - ~~Pathfinding~~
  - ~~Levels/Maps~~
  - Scoring
  - Special Zombies