Do you have ranged zombies yet?

- Last week we added ranged zombies, and for starters let's finish them off.

- So… open up the slides and project from last week and continue where you left off!

github.com/JamesAdey/zompy/blob/master/slides/cc_week7.pdf

- Alternatively, if you've done this already, free play for a while!

Creative Computing. Week 8

(cool computers with stuffing)

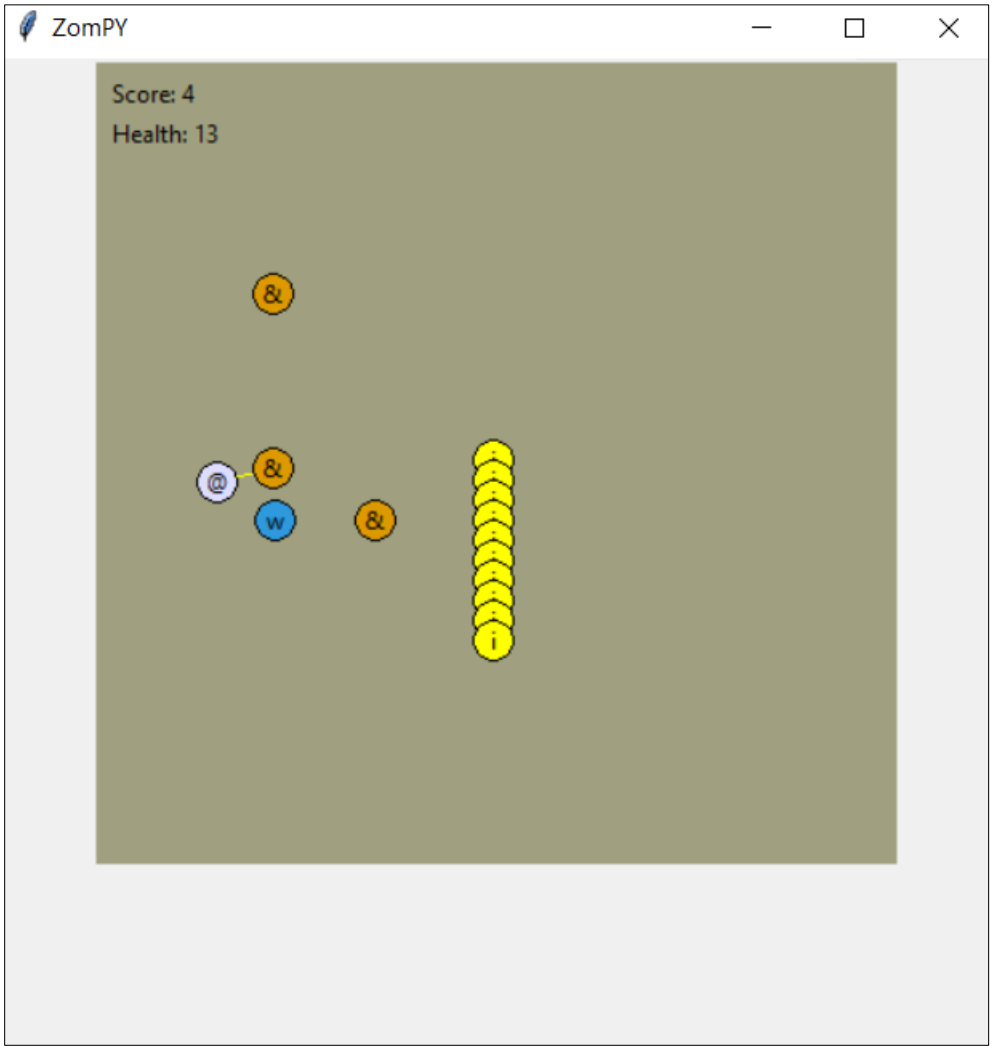
If you don't have the project yet...

- I've hosted the basic code on my GitHub

- Go to: ***github.com/jamesadey/zompy/tree/week8***
  - *This is the project containing finished code from week 7, ready for week 8.*
- 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…

The finished* game.

- Our zombies can move… intelligently!
- And they can be shot…
- We have spawners…
- And item pickups!
- With random maps!
- UI and some very special zombies…

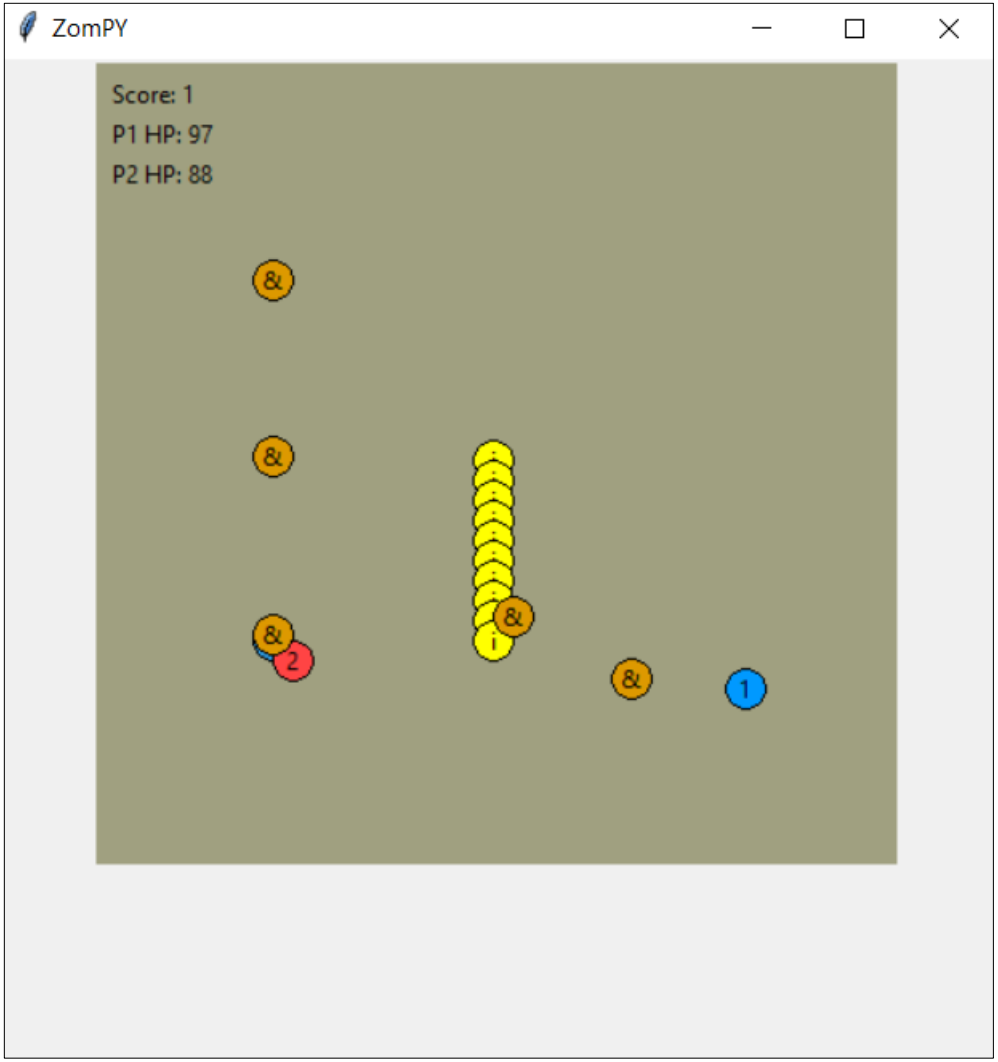


**not really*

By special request… Multiplayer!

- Our zombies can move… intelligently!
- And they can be shot…
- We have spawners…
- And item pickups!
- With random maps!
- UI and some very special zombies…

Now we're adding… a friend?



Local Multiplayer.

(modifying the project
to work with a second player)

Multiplayer Differences

- So far we've had a single variable used to denote our player.
- Networked multiplayer games often use arrays of player objects, each one tied to a different game client and referring to different instances of a player class.

- For our game we will now move to using a list of players, as opposed to a single global variable.
  - This should allow for many players to be added, not just 2.

Under-the-hood changes

- Just in case you've modified the engine (or were just curious), here's what I've changed….
- The engine code for input reading has changed to reading the key symbol
  - This allows for arrow keys to be used!
- The navgrid now works with lists of players.
- Collisions also work with lists of players.

The Fun Part.

(actually coding stuff)

Download these slides.

github.com/JamesAdey/zompy/blob/master/slides/cc_week8.pdf

You'll need to copy these 5 files.

- I've changed the core engine & systems, now you need to edit the rest.
- So first **Update** these 5 files as they contain new code and modifications.

github.com/JamesAdey/zompy/blob/week8/engine.py

github.com/JamesAdey/zompy/blob/week8/game_manager.py

github.com/JamesAdey/zompy/blob/week8/item_manager.py

github.com/JamesAdey/zompy/blob/week8/navgrid.py

github.com/JamesAdey/zompy/blob/week8/gameworld.py

- Then follow the instructions in the following slides.

Edit: *player.py*

- ***Add variables*** *to hold the players input and identity information*

```
class Player(GameObject):

    playerNum = None
    upKey = "w"
    downKey = "s"
    leftKey = "a"
    rightKey = "d"

    health = 100
    radius = 10
```

Swap the constants for variables here

```
shoot = False

if(gameGlobals.is_key_pressed(self.leftKey)):
    xMove -=1

if(gameGlobals.is_key_pressed(self.rightKey)):
    xMove +=1

if(gameGlobals.is_key_pressed(self.upKey)):
    yMove -=1

if(gameGlobals.is_key_pressed(self.downKey)):
    yMove +=1

if(gameGlobals.is_mouse_pressed("left")):
```

- *Then change the* ***do_inputs()*** *function to use the new variables.*

- *Modify the* ***update()*** *function so that it sends the playerNum when updating the health UI*

```
# update our status in the GUI
if(self.hurt):
    gameGlobals.gameManager.update_player_health(self.playerNum, self.health)
    self.hurt = False
```

Keep Editing: *player.py*

- ***Create a new function*** *called* ***set_player_info()****, to setup the variables for the player*

```
def set_player_info(self,num,upKey,downKey,leftKey,rightKey):
    self.playerNum = num
    self.upKey = upKey
    self.downKey = downKey
    self.leftKey = leftKey
    self.rightKey = rightKey
```

Edit: *ranged_zombie.py*

- *Our ranged zombie only attacked 1 player…*
- *We need to attack all players if possible. So add a loop to try and attack each one.*

- *Inside the **update()** function, swap the single method call for a loop.*

```
self.move(targetX,targetY)

for player in gameGlobals.players:
    self.attack(gameGlobals,player)

if(self.health <= 0):
```

Edit: *zompy_globals.py*

- *Remove the **player** variable, and replace it with a list of **players***

```
class ZompyGlobals(GameGlobals):
    zoms = 10
    bulletManager = None
    gameManager = None
    itemManager = None
    gameWorld = None
    navGrid = None
    # list of players
    players = []
```

- *Edit **setup_ui()**, copy the health text gui, change the text and move it down a bit.*

```
super().add_game_object(scoreText)

# create a health gui and link it to the game manager
healthText = GUIText(x=10,y=30,baseText="P1 HP: ")
gGlobals.gameManager.set_health_gui(1, healthText)
healthText.set_text("100")
super().add_game_object(healthText)

healthText = GUIText(x=10,y=50,baseText="P2 HP: ")
gGlobals.gameManager.set_health_gui(2, healthText)
healthText.set_text("100")
super().add_game_object(healthText)
```

- *Don't forget to pass in the player number it belongs to!*

Edit: *Your levels.*

- *In our levels, we need to replace the single "create player" block, with a 2 blocks, and remember to set the player's inputs.*

```
# ALWAYS remember to create the players!
p1 = Player(x=300,y=300,char='1',colour="#0099FF")
p1.set_player_info(1,"w","s","a","d")
gameEngine.add_game_object(p1)
gameGlobals.players.append(p1)

p2 = Player(x=100,y=300,char='2',colour="#FF4444")
p2.set_player_info(2,"Up","Down","Left","Right")
gameEngine.add_game_object(p2)
gameGlobals.players.append(p2)
```

Now test!

- Yes, the players do shoot at the same place.
  - We don't have 2 mice connected, so it would be a bit difficult to make them shoot independently.
- That said.. This could provide some interesting gameplay, where both players need to collaborate to stay alive.
- Alternatively, you could add a different weapon for player 2…
  - Next we will add a melee weapon to player 2, using overlapping circles instead of line tracing.

Edit: *player.py*

- *Add some variables to store our attacking capabilities*

```
leftKey = "a"
rightKey = "d"
meleeKey = "space"
canShoot = True
canMelee = True

health = 100
```

- *We're going to refactor the **update()** function, copying the shooting code into new **shoot()** method.*

```
def shoot(self, gameGlobals):
    # get the mouse position as the end of our bullet
    (endX,endY) = gameGlobals.get_mouse_position()
    # trace the physics line against the zombies
    hit = gameGlobals.gameWorld.traceline(self.x,self.y,endX,endY)
    if(isinstance(hit,zombie.Zombie)):
        hit.take_damage(10)
        # override the end position to draw the bullets
        endX,endY = hit.get_position()
    gameGlobals.bulletManager.fire_bullet(self.x,self.y,endX,endY)

def do_inputs(self, gameGlobals):
```

- *Now make a **melee()** method as follows:*

```
def melee(self, gameGlobals):
    # overlap a circle against the zombies
    hit = gameGlobals.gameWorld.overlap_circle(self.x,self.y,25)
    if(isinstance(hit,zombie.Zombie)):
        hit.take_damage(30)
        # show a bullet tracer to simulate the attack
        endX,endY = hit.get_position()
        gameGlobals.bulletManager.fire_bullet(self.x,self.y,endX,endY)

def shoot(self, gameGlobals):
```

Edit: *player.py*

- *We need to edit the **do_inputs()** function, to permit melee attacks.*

```
def do_inputs(self, gameGlobals):
    xMove = 0
    yMove = 0
    shoot = False
    melee = False

    if(gameGlobals.is_key_pressed(self.leftKey)):
        xMove -=1

    if(gameGlobals.is_key_pressed(self.rightKey)):
        xMove +=1

    if(gameGlobals.is_key_pressed(self.upKey)):
        yMove -=1

    if(gameGlobals.is_key_pressed(self.downKey)):
        yMove +=1

    if(gameGlobals.is_mouse_pressed("left") and self.canShoot):
        shoot = True

    if(gameGlobals.is_key_pressed(self.meleeKey) and self.canMelee):
        melee = True

    return (xMove,yMove,shoot,melee)
```

Edit: *player.py*

- *We should **modify the update() function** so that it calls the appropriate attack code.*

```
def update(self, gameGlobals):
    # get the inputs from the keyboard
    (xMove,yMove,shoot,melee) = self.do_inputs(gameGlobals)
    # do the movement
    self.x = self.x + xMove
    self.y = self.y + yMove
    if(melee):
        self.melee(gameGlobals)
    if(shoot):
        self.shoot(gameGlobals)
    # update our status in the GUI
    if(self.hurt):
        gameGlobals.gameManager.update_player_health(self.playerNum, self.health)
        self.hurt = False
```

- *And add some more parameters to **set_player_info()***

```
def set_player_info(self,num,upKey,downKey,leftKey,rightKey,meleeKey,canMelee,canShoot):
    self.playerNum = num
    self.upKey = upKey
    self.downKey = downKey
    self.leftKey = leftKey
    self.rightKey = rightKey
    self.meleeKey = meleeKey
    self.canMelee = canMelee
    self.canShoot = canShoot
```

Finally, edit your levels.

- *We now need to **pass the correct parameters** when setting our player info:*

```
# ALWAYS remember to create the players!
p1 = Player(x=300,y=300,char='1',colour="#0099FF")
p1.set_player_info(1,"w","s","a","d","q",False,True)
gameEngine.add_game_object(p1)
gameGlobals.players.append(p1)

p2 = Player(x=100,y=300,char='2',colour="#FF4444")
p2.set_player_info(2,"Up","Down","Left","Right","space",True,False)
gameEngine.add_game_object(p2)
gameGlobals.players.append(p2)
```

- That was a lot of modifications… Hopefully the players will now have different attacks!

Now what?

- I've got nothing left for you. So it's time for some **FREE PLAY**!

- You can add anything else you want into the game.
  - *Try looking at past slides and having a go at the extensions.*

Well… That's all folks.

- Sadly, this is the end of the zompy project, and I thought the project went great.
- You now have a whole game prototype and can take it wherever you want. Add more features, or don't, that's entirely up to you!

- Any final thoughts and feedback?