| Input | Process | Output |
| --- | --- | --- |
| * Keyboard inputs (arrow keys for movement, space for auto-aim shooting) * Mouse position tracking for aiming * Mouse clicks for manual shooting * Special key triggers (T for teleport, B for beam weapon, C for spread shots) * Game state inputs (R to restart, ESC to quit) | * Game loop (run method) * Physics calculations for player and enemy movement * Collision detection using pygame.colliderect and vector dot product for beam (bullets-enemies, player-enemies, spikes-entities) * Particle system updates and animations, applying gravity to explosions, otherwise propagating velocity whilst introducing fade * Randomised enemy spawning on periphery * Powerup random generation and management to the variables in game class, i.e teleport count. * Level progression system, reducing player speed, increasing enemy spawn rate and speed, reducing spike spawn interval * Collision damage * Xp tracking with exponential scaling. | -Rendering of game elements(  -Particles including directed trails, recursive explosions, and flames over cursor.  -Health, XP bar, powerup count indicators on the screen.  -Warning indicators prior to spike spawning  -Game over screen  -Level up screen(continuous and does not pause the game)  -Widened range of player capability to teleport or destroy more enemies than usual.  -A more intriguing playing experience |

### **Pseudocode**

CLASS Player

FUNCTION Initialize(x, y, assets)

SET x = x

SET y = y

SET vx = 0

SET vy = 0

SET speed = PLAYER\_SPEED from app

SET animations = assets["player"]

SET beam = assets["beam"][0]

SET beam\_display = [0, 0, 0, 0, FALSE]

SET state = "idle"

SET frame\_index = 0

SET animation\_timer = 0

SET animation\_speed = 8

SET xp = 0

SET image = animations[state][frame\_index]

SET rect = create rectangle from image centered at (x, y)

SET facing\_left = FALSE

SET health = 5

SET bullet\_speed = 10

SET bullet\_size = 10

SET bullet\_count = 1

SET shoot\_cooldown = 20

SET shoot\_timer = 0

SET bullets = empty list

END FUNCTION

FUNCTION add\_xp(amount)

INCREMENT xp by amount

END FUNCTION

FUNCTION handle\_input()

GET pressed keys

SET vel\_x = 0, vel\_y = 0

IF 'a' key is pressed THEN

DECREMENT vel\_x by speed

END IF

IF 'd' key is pressed THEN

INCREMENT vel\_x by speed

END IF

IF 's' key is pressed THEN

INCREMENT vel\_y by speed

END IF

IF 'w' key is pressed THEN

DECREMENT vel\_y by speed

END IF

INCREMENT x by vel\_x

INCREMENT y by vel\_y

SET x = maximum of (0, minimum of (x, WIDTH))

SET y = maximum of (0, minimum of (y, HEIGHT))

SET rect.center = (x, y)

IF vel\_x is not 0 OR vel\_y is not 0 THEN

SET state = "run"

ELSE

SET state = "idle"

END IF

IF vel\_x < 0 THEN

SET facing\_left = TRUE

ELSE IF vel\_x > 0 THEN

SET facing\_left = FALSE

END IF

END FUNCTION

FUNCTION update()

FOR EACH bullet IN bullets

CALL bullet.update()

IF bullet.y < 0 OR bullet.y > HEIGHT OR bullet.x < 0 OR bullet.x > WIDTH THEN

REMOVE bullet from bullets

END IF

END FOR

INCREMENT animation\_timer by 1

IF animation\_timer >= animation\_speed THEN

SET animation\_timer = 0

SET frames = animations[state]

SET frame\_index = (frame\_index + 1) modulo length of frames

SET image = frames[frame\_index]

SET center = rect.center

SET rect = create rectangle from image

SET rect.center = center

END IF

END FUNCTION

FUNCTION draw(surface)

IF facing\_left THEN

SET flipped = flip image horizontally

DRAW flipped onto surface at rect position

ELSE

DRAW image onto surface at rect position

END IF

FOR EACH bullet IN bullets

CALL bullet.draw(surface)

END FOR

END FUNCTION

FUNCTION take\_damage(amount)

SET health = maximum of (0, health - amount)

END FUNCTION

FUNCTION shoot\_toward\_position(tx, ty)

IF shoot\_timer >= shoot\_cooldown THEN

RETURN

END IF

SET dx = tx - x

SET dy = ty - y

SET dist = square root of (dx² + dy²)

IF dist equals 0 THEN

RETURN

END IF

SET vx = (dx / dist) × bullet\_speed

SET vy = (dy / dist) × bullet\_speed

SET angle\_spread = 10

SET base\_angle = arctangent of (vy, vx)

SET mid = (bullet\_count - 1) / 2

FOR i = 0 TO bullet\_count - 1

SET offset = i - mid

SET spread\_radians = angle\_spread × offset in radians

SET angle = base\_angle + spread\_radians

SET final\_vx = cosine of angle × bullet\_speed

SET final\_vy = sine of angle × bullet\_speed

CREATE new Bullet at (x, y) with velocity (final\_vx, final\_vy) and size bullet\_size

ADD bullet to bullets list

END FOR

SET shoot\_timer = 0

END FUNCTION

FUNCTION circleshot()

SET num = 15

SET deg = 360/num in radians

FOR x = 0 TO num - 1

CREATE new Bullet at (x, y) with velocity (

cosine of (x × deg) × bullet\_speed × 0.5,

sine of (x × deg) × bullet\_speed × 0.5

) and size bullet\_size

ADD bullet to bullets list

END FOR

END FUNCTION

FUNCTION shoot\_toward\_mouse(pos)

SET mx, my = pos

CALL shoot\_toward\_position(mx, my)

END FUNCTION

FUNCTION shoot\_toward\_enemy(enemy)

CALL shoot\_toward\_position(enemy.x, enemy.y)

END FUNCTION

FUNCTION teleport(posx, posy)

SET x = posx

SET y = posy

END FUNCTION

FUNCTION shootbeam(posx, posy)

SET angle\_rad = arctangent of (posy - y, posx - x)

SET angle\_deg = angle\_rad in degrees

SET initial\_beam\_width = 25

SET beam\_start\_x = x + (cosine of angle\_rad × 470)

SET beam\_start\_y = y + (sine of angle\_rad × 470)

SET beam\_display = [

angle\_deg,

beam\_start\_x,

beam\_start\_y,

30,

TRUE

]

END FUNCTION

END CLASS