Feature Specifications – Enemies, Aiming & Shooting, Player Movement

1. Enemies

1.1 Functionality in stage 2

- Enemies are placed in the level, they are **not** spawned during runtime
- Enemies also attack when they are off-screen (i.e. when the player is not looking at them during combat)
- Enemies have health bars attached that are shown once they take damage
- Enemies regenerate health when returning to their patrol state
- Standard enemy type
  - Patrols within a three-dimensional area assigned to them
    - After a delay (random duration between two assigned integers), it receives a new objective point within the area and moves towards it
    - The delay for the next objective point only starts on reaching the current objective
  - Once they detect the player, they start attacking
    - The player is detected if they enter the enemy's three-dimensional cone of vision **and** the line of sight between enemy and player is **not** interrupted by collision
    - The player is detected if they hit the enemy with any arrow
  - Returns to the idle state if the player does **not** fulfill any of the two detection conditions for a while
  - Shoots X amount of projectiles after a short wind-up to attack
    - Small delay between each of the shots
    - Wind-up -> projectile 1 -> delay -> projectile 2 -> delay -> … -> projectile X -> longer delay -> repeat
    - Each shot has a random offset from the direction drone-to-player (on the yaw axis), giving it a chance to miss
    - Amount of shots per attack loop needs to be tested
  - Keeps patrolling movement pattern even in combat, does **not** follow the player
- Shotgun enemy type
  - Basic version should be implemented for stage 2 already
  - Still needs to be specified

1.2 Exposed variables

- Standard Enemy
  - Movement
    - Movement boundaries (3D box)
      Speed
      Minimum delay duration on reaching objective point

Maximum delay duration on reaching objective point

Detection

Depth of initial cone of vision
Radius of initial cone of vision
Depth of cone of vision on player detection
Radius of cone of vision on player detection
Reset timer

Health

Maximum health points
Regeneration tick interval
Health regeneration per tick

Attacks

Attack wind-up duration
Amount of bullets per attack loop
Delay between bullets of the same attack loop
Maximum shot offset from player direction
Bullet speed
Damage per bullet
Rotation speed to track the player

1.3 Ideas for future stages

Enemies respawn on player death

Damage numbers appear on hitting an enemy with an arrow

Sound effects let players know when enemies are winding up their shots off-screen

Shotgun enemy type

Rapidly moves towards the player

Fires one powerful shot when it reaches a close distance to the player

Gets knocked back from their own shot

Player does **not** get knocked back (because it would be disorienting in first-person)

Follows the player within the boundaries of a large area

Boss will be the third and last enemy type for this game

2. Aiming & Shooting

2.1 Functionality in stage 2

Holding LMB draws the bow, releasing LMB fires the arrow

Draw time affects the speed of the arrow and thus the impact of gravity on it

Crosshair resembles drawing the bow and its impact on the ‘arrow drop’

Could be tall rectangle with the lower part closing in as you draw the bow

Draw time also affects the damage dealt by the arrow

Linearly interpolates between a minimum and maximum damage based on the draw time

Distance of the enemy or speed of the arrow on impact is **not** relevant for damage

Damage example

2 secs draw time, enemy 2 meters away = 25 damage

2 secs draw time, enemy 4 meters away = 25 damage

4 secs draw time, enemy 2 meters away = 50 damage

Can only draw bow to certain degree (cap)

After reaching this cap, the speed of the arrow on release and the damage dealt **no** longer increase

There are **no** spray patterns or random offsets for shot arrows

There is **no** limit on basic arrows, they are an infinite resource

No HUD element for ammunition

2.2 Exposed variables

All exposed variables should sit on the parent object of the player or the main camera

Arrows do not have exposed variables but should be instantiated with the correct values

Aiming

Mouse sensitivity

Shooting

Shooting cooldown
Maximum draw duration
Minimum arrow speed
Maximum arrow speed
Minimum arrow damage
Maximum arrow damage
Minimum arrow gravity
Maximum arrow gravity

2.3 Formulas

Arrow speed = minSpeed + ((drawTime[0, maxDrawTime] / maxDrawTime) * (maxSpeed - minSpeed))

Example: minSpeed = 40, maxSpeed = 100, maxDrawTime = 4, drawTime = 1

Arrow speed = 40 + (1 / 4) * (100 - 40) = 40 + 0,25 * 60 = **55**

Arrow damage = minDamage + ((drawTime[0, maxDrawTime] / maxDrawTime) * (maxDamage - minDamage))

Example: minDamage = 150, maxDamage = 300, maxDrawTime = 4, drawTime = 4

Arrow damage = 150 + (4 / 4) * (300 - 150) = 150 + 1 * 150 = **300**

Arrow gravity = maxGravity - ((drawTime[0, maxDrawTime] / maxDrawTime) * (maxGravity - minGravity))

Example: minGravity = 30, maxGravity = 80, maxDrawTime = 4, drawTime = 0,5

Arrow gravity = 80 - (0,5 / 4) * (80 - 30) = 80 - 0,125 * 50 = **73,75**

2.4 Ideas for future stages

Arrows have a trail effect attached (so you can tell where they went)

Arrows stick to solid surfaces they hit (enemies, walls, …)

Purely cosmetic, **no** actual collision

Amount of visible arrows at a time is capped (for performance reasons)

3. Player Movement

3.1 Functionality in stage 2

WASD is used for basic movement

Arrow keys are **not** used

SPACE launches a jump

You can dash mid-air

You can aim and shoot mid-air

You can jump for a short timeframe when leaving the ground (Coyote jump)

Does **not** work when a jump was used to leave the ground in the first place

SHIFT performs a dash

Boost in velocity in the direction the player is moving in

No effect if player is stationary

You can aim and shoot while dashing

Short cooldown to limit use of dash

3.2 Exposed variables

Basics

Maximum speed
Acceleration rate
Enable deceleration?
Deceleration rate
Maximum walkable slope angle

Jump

Jump force
Default gravity
New gravity on reaching jump peak
Coyote jump timeframe

Dash

Dash force
Cooldown duration

3.3 Ideas for future stages

Bow lights up during dash (either in approval or disapproval depending on narrative)

No additional movement features

Will be outsourced into special quivers / arrow types