# Enemy

|  |
| --- |
| 1. **void** Update() 2. { 3. transform.position = Vector2.MoveTowards(transform.position, moveSpot.position, speed \* Time.deltaTime); 5. **if** (Vector2.Distance(transform.position, moveSpot.position) < 0.2f) 6. { 7. **if** (waitTime <= 0) 8. { 9. waitTime = startWaitTime; 10. **if** (GameObject.FindGameObjectWithTag("Player")) 11. { 12. **int** trace = Random.Range(0, 2); 13. **if** (trace == 1) 14. { 15. Debug.Log($"enemy {name} is tracing player"); 16. moveSpot.position = GameObject.FindGameObjectWithTag("Player").GetComponent<Transform>().position; 17. } 18. **else** 19. { 20. Debug.Log($"enemy {name} is ignoring player"); 21. moveSpot.position = **new** Vector2(Random.Range(min\_X, max\_X), Random.Range(min\_Y, max\_Y)); 22. } 23. } 24. **else** 25. { 26. moveSpot.position = **new** Vector2(Random.Range(min\_X, max\_X), Random.Range(min\_Y, max\_Y)); 27. } 28. } 29. **else** 30. { 31. waitTime -= Time.deltaTime; 32. } 33. } 35. } |
| Enemy movement behaviour  Enemy has 2 phases: move and stop  Move phase:  When there is a player exist in the playground, one enemy has 50% chance (D2) of targeting the player, or just move to random location within the border, then take a rest for an amount of time.  If there is an situation that there is no player, for example, player is dead, enemies then just moving around randomly.  Stop phase:  Count down it’s timer, then move again. |

|  |
| --- |
| 1. **void** Update() 2. { 3. **if** (isBeingAttacked) 4. { 5. **if** (name == "Skull") 6. { 7. ScoreSpript.scoreValue += 1; 9. } 10. **else** **if** (name == "Ghost") 11. { 12. ScoreSpript.scoreValue += 2; 13. } 14. **else** **if** (name == "Demon") 15. { 16. ScoreSpript.scoreValue += 3; 17. } 18. **else** **if** (name == "Clown") 19. { 20. ScoreSpript.scoreValue += 4; 21. } 22. **else** **if** (name == "Health") 23. { 25. **if** (Player.currentHealth < 5) 26. { 27. Player.currentHealth += 1; 28. } 29. **else** 30. Player.currentHealth = 5; 32. } 33. Destroy(gameObject); 34. } 35. } |
| Enemies give score based on their name when they are being attacked  “Health” is an enemy object but it heal player instead of dealing damage |
| 1. **void** OnTriggerEnter2D(Collider2D hitInfo) 2. { 3. Player target = hitInfo.GetComponent<Player>(); 4. **if** (target != **null**) 5. { 6. **if** (name == "Health") 7. { 8. FindObjectOfType<AudioManager>().Play("Get Health"); 9. target.Heal(1); 10. } 11. **else** **if** (name == "Clown") 12. { 13. target.TakeDamage(3); 14. } 15. **else** **if** (name == "Demon") 16. { 17. target.TakeDamage(2); 18. } 19. **else** **if** (name == "Ghost") 20. { 21. target.TakeDamage(1); 22. } 23. **else** **if** (name == "Skull") 24. { 25. target.TakeDamage(1); 26. } 27. Destroy(gameObject); 28. } 29. } |
| Enemy deal damage to player via collision, their damage is based on their name. |

|  |
| --- |
| 1. // Update is called once per frame 2. **void** Update() 3. { 4. heartsCoolDown += Time.deltaTime; 5. **if** (heartsCoolDown >= timeBetweenHearts) 6. { 7. heartsCoolDown = 0; 8. **if** (GameObject.FindGameObjectWithTag("Heart") == **null**) 9. { 10. **int** chance = Random.Range(0, 3); 11. Debug.Log(chance); 12. **if** (chance == 0) 13. { 14. SpawningHeart(); 15. } 16. } 17. } 18. } |
| There are 25% chance to spawn a heart for every “heartCoolDown” amount of time, one heart can be existed at a time. |

# Player

|  |
| --- |
| 1. **void** Update() 2. { 3. **if** (isBeingHeld == **true**) 4. { 5. Vector3 mousePos; 6. mousePos = Input.mousePosition; 7. mousePos = Camera.main.ScreenToWorldPoint(mousePos); 9. **float** top = borderTopRight.position.x; 10. **float** bottom = borderDownLeft.position.x; 11. **float** left = borderDownLeft.position.y; 12. **float** right = borderTopRight.position.y; 13. **if** (mousePos.x <= top && mousePos.x >= bottom && mousePos.y >= left && mousePos.y <= right) 14. { 15. **this**.gameObject.transform.localPosition = **new** Vector3(mousePos.x - startPosX, mousePos.y - startPosY); 16. } 17. **else** 18. { 19. isBeingHeld = **false**; 20. } 21. } 22. } |
| Player can drag the character around the screen with mouse, as long as the character is not over the designed border. |

# Stages

|  |
| --- |
| 1. **void** SpawningEnemy() 2. { 3. //D20 for enemies 4. **int** chance = Random.Range(0, 19); 5. Debug.Log(chance); 6. Transform \_sp = spawnPoints[Random.Range(0, spawnPoints.Length)];  9. **int** arrayIndex; 10. **if** (chance <2) 11. { 12. arrayIndex = 0; 13. } 14. **else** **if** (chance >= 2 && chance<6) 15. { 16. arrayIndex = 1; 17. } 18. **else** **if** (chance >=6 && chance <11) 19. { 20. arrayIndex = 2; 21. } 22. **else** 23. { 24. arrayIndex = 3; 25. } 27. Sprite enemySprite = enemySprites[arrayIndex]; 28. **string** enemyName = enemySprite.name; 30. GameObject newEnemy = Instantiate(enemyPrefab, \_sp.position, \_sp.rotation); 31. newEnemy.name = enemyName; 32. newEnemy.GetComponent<SpriteRenderer>().sprite = enemySprite; 33. newEnemy.GetComponent<EnemyPatrol>().speed = **this**.enemySpeed;  36. } |
| Make a D20 for spawning enemies  Make a D4 for spawn point  Assign the random number from D20 to a specific Sprite  Make an enemy with the assigned sprite, sprite name  Spawn an enemy in the location from a random chosen spawn point |
| 1. **void** WaveCompleted() 2. { 3. Debug.Log("Wave completed"); 4. FindObjectOfType<AudioManager>().Play("Next Level"); 5. FindObjectOfType<CountdownTimer>().ResetTimer(); 6. state = SpawnState.COUNTING; 7. waveCountDown = timeBetweenWave; 9. **if** (nextWave +1 > waves.Length -1) 10. { 11. nextWave = 0; 12. **this**.enemySpeed += 3; 13. Debug.Log("Looping waves"); 14. } 15. **else** 16. { 17. **this**.enemySpeed += 2; 18. nextWave++; 19. } 21. } |
| Level looping, reset the timer and increase enemy speed |

|  |
| --- |
| 1. **void** Update() 2. { 3. **if** (currentTime <= 0) 4. { 5. Destroy(GameObject.FindGameObjectWithTag("Player")); 6. } 7. **if** (GameObject.FindGameObjectWithTag("Player") != **null**) 8. { 9. ReturnTicks(); 10. timer.SetText(currentTime.ToString("0")); 11. } 12. **else** 13. { 14. **return**; 15. } 16. } |
| Player has to complete a stage in an amount of time, or being destroyed! |

|  |
| --- |
| 1. **void** Update() 2. { 3. **if** (Ended()) 4. { 5. endLevelUI.SetActive(**true**); 6. } 7. }  10. **bool** Ended() 11. { 12. **if** (GameObject.FindGameObjectWithTag("Player") == **null**) 13. { 14. **return** **true**; 15. } 16. **return** **false**; 17. } |
| When player is no longer exist (destroyed by timer or by enemies) the game is over. |