# Probability and Utility Worksheet and Checklist

## Probability Distributions

|  |
| --- |
| Enemy spawn – there are four enemies within the game, the outcome of the next monster to spawn is random. |
| Health – Hearts are spawned throughout the game will be random on a % and timer |
| Spawn Location – The outcome of the random spawn location of the enemy |

|  |  |
| --- | --- |
| **Outcome** | **Probability** |
| Enemies will have a different value in score, which will reflect on their value of spawn rate. Assume value: Skull – 1 point, Ghost – 2 points, Demon – 3 points and Clown – 4 points  **Before testing** | Skull: 0.25  Ghost: 0.25  Demon: 0.25  Clown: 0.25  Total: 1 |
| Enemies will have a different value in score, which will reflect on their value of spawn rate. Assume value: Skull – 1 point, Ghost – 2 points, Demon – 3 points and Clown – 4 points  **Final rate** | Skull: 0.50  Ghost: 0.25  Demon: 0.15  Clown: 0.10  Total: 1 |
| Chance of spawning power-up at every quarter of the game time(60 seconds) | Heart: 0.25 x 15 secs  Heart: 0.25 x 15 secs  Heart: 0.25 x 15 secs  Heart: 0.25 x 15 secs  Total: 1 |
| Spawn Locations would use either 4 sides of the grid. Probability would be equal since no side would change the difficulty | Top: 0.25  Left: 0.25  Right: 0.25  Bottom: 0.25  Total: 1 |

**Game Code/ Algorithm :**

|  |
| --- |
| **void** Update()  {      transform.position = Vector2.MoveTowards(transform.position, moveSpot.position, speed \* Time.deltaTime);  **if** (Vector2.Distance(transform.position, moveSpot.position) < 0.2f)      {  **if** (waitTime <= 0)          {              waitTime = startWaitTime;  **if** (GameObject.FindGameObjectWithTag("Player"))              {  **int** trace = Random.Range(0, 2);  **if** (trace == 1)                  {                      Debug.Log($"enemy {name} is tracing player");                      moveSpot.position = GameObject.FindGameObjectWithTag("Player").GetComponent<Transform>().position;                  }  **else**                  {                      Debug.Log($"enemy {name} is ignoring player");                      moveSpot.position = **new** Vector2(Random.Range(min\_X, max\_X), Random.Range(min\_Y, max\_Y));                  }              }  **else**              {                  moveSpot.position = **new** Vector2(Random.Range(min\_X, max\_X), Random.Range(min\_Y, max\_Y));              }          }  **else**          {              waitTime -= Time.deltaTime;          }      }    } |
| Enemy movement behavior  The 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 a random location within the border, then take a rest for an amount of time.  If there is a situation that there is no player, for example, the player is dead, enemies then just moving around randomly.  Stop phase:  Count down its timer, then move again. |
| **void** Update()  {  **if** (isBeingAttacked)      {  **if** (name == "Skull")          {              ScoreSpript.scoreValue += 1;            }  **else** **if** (name == "Ghost")          {              ScoreSpript.scoreValue += 2;          }  **else** **if** (name == "Demon")          {              ScoreSpript.scoreValue += 3;          }  **else** **if** (name == "Clown")          {              ScoreSpript.scoreValue += 4;          }  **else** **if** (name == "Health")          {    **if** (Player.currentHealth < 5)              {                  Player.currentHealth += 1;              }  **else**                  Player.currentHealth = 5;          }          Destroy(gameObject);      }  } |
| 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 |
| // Update is called once per frame  **void** Update()  {      heartsCoolDown += Time.deltaTime;  **if** (heartsCoolDown >= timeBetweenHearts)      {          heartsCoolDown = 0;  **if** (GameObject.FindGameObjectWithTag("Heart") == **null**)          {  **int** chance = Random.Range(0, 3);              Debug.Log(chance);  **if** (chance == 0)              {                  SpawningHeart();              }          }      }  } |
| There are 25% chance to spawn a heart for every “heartCoolDown” amount of time, one heart can be existed at a time. |
| **void** SpawningEnemy()  {      //D20 for enemies  **int** chance = Random.Range(0, 19);      Debug.Log(chance);      Transform \_sp = spawnPoints[Random.Range(0, spawnPoints.Length)];      **int** arrayIndex;  **if** (chance <2)      {          arrayIndex = 0;      }  **else** **if** (chance >= 2 && chance<6)      {          arrayIndex = 1;      }  **else** **if** (chance >=6 && chance <11)      {          arrayIndex = 2;      }  **else**      {          arrayIndex = 3;      }        Sprite enemySprite = enemySprites[arrayIndex];  **string** enemyName = enemySprite.name;        GameObject newEnemy = Instantiate(enemyPrefab, \_sp.position, \_sp.rotation);      newEnemy.name = enemyName;      newEnemy.GetComponent<SpriteRenderer>().sprite = enemySprite;      newEnemy.GetComponent<EnemyPatrol>().speed = **this**.enemySpeed;      } |
| 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 |

## Utility

|  |  |
| --- | --- |
| **Outcome:** | **Value:** |
| Each monster is assigned a point value associated with their difficulty aspect | Skull: 1  Ghost: 2  Demon: 3 Clown: 4 |
| A power-up with hearts will add health to the player | Health: +1 heart/hp |
| The enemy coming into contact with character, reduction in health  **Prior to testing** | Health: -1 heart for all enemies |
| The enemy coming into contact with character, reduction in health  **Final** | Skull: -1 heart  Ghost: -1 heart  Demon: -2 hearts  Clown: -3 hearts |
| Increasement in monsters spawning within each level | Level 1: 15  Level 2: 20  Level 3: 25  Level 4: 30 |

|  |
| --- |
| *Expected utility of monsters in points: 0.50 \* 1 + 0.25 \* 2 + 0.15 \* 3 + 0.10 \* 4 = 1.85*  *Expected utility of monsters (spawn each level): 15 + 5 + 5 + 5 to infinite*  *Health: 0.25 \* 15 = 3.75*  *This suggests if the player can score above 2 points per second, they are on track to beat the current level. We can add to the difficulty with each level by increasing the speed of the enemy. The level design is informed by adjusting the utility of various sources in the game which includes monsters per level, speed increasement, and power-up. The availability of a power-up throughout the game, as well as the different value of deduction each monster, will ultimately come down to the player’s skill and speed.* |