Spawner Guide for balancing

overview of script:

Variables:

```
private int startingMoney;                                  //Money at the start of the game day
private int EndMoney;                                       //Money at the end of the game day
private float dayRating = 1f;                               //The float day rating that changes the spawn system
private float vendorRating;                                 //SubRating of the day rating relating to the vendors visitors
private float econRating;                                   //SubRating of the day rating relating to the daily profits
private float trackRating;                                  //SubRating of the day rating relating to the track quality
private int totTrackScore;                                  //Integer containing the total tack score in scene
private int prevTrackScore = 0;                             //Integer containing the track score from the previous day
private float progRating;                                   //Float containing the overall progression rating
private float avgNumVendors;                                //Float containing the average number of vendors in scene
private int totalDayVisitors;                               //Integer containing the total visitors for a day
private float DayStarRating;                                //Converted day rating to be displayed as a star rating
private int numTracks;                                      //Contains the number of tracks in scene
private int numDeco;                                        //Contains the number of decorations in scene
public GameObject[] spawnPoints;                            //List containing all possible spawn points for visitors
public GameObject[] raceManagers;                           //List containing gameobjects for all start track prefabs in scene
public HandleDayNightCycle time;                            //Reference to the day night cycle script
public float overallSatisfaction = 1f;                      //Float containing the overall visitor satisfaction
[EventfulProperty] private int _visitorsIn = 0;             //Integer containing the current number of visitors
public List<GameObject> prefabVisitors = new List<GameObject>();  //List containing all visitor prefabs
System.Random rand = new System.Random();                   //Reference to random, used for random integers
```

No changes will need to be made here but this gives an overview of what they do.

Functions:

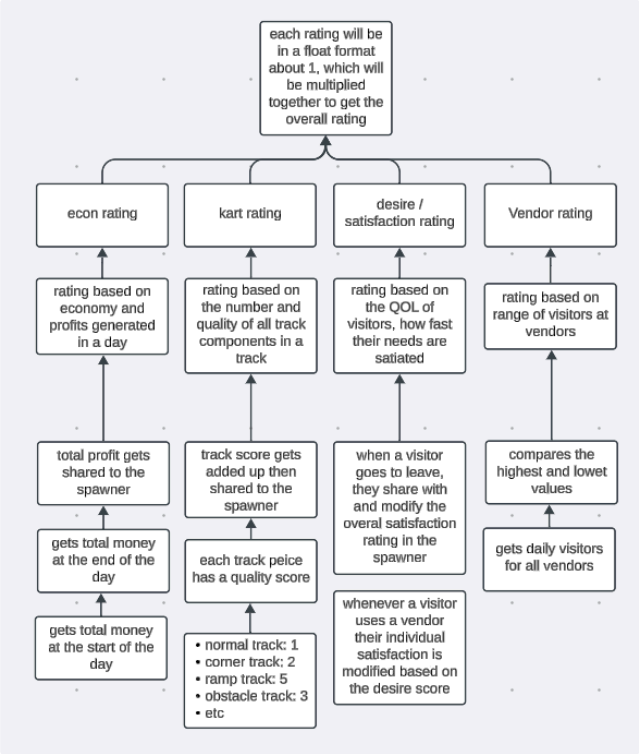
```
Unity Message | 0 references
void Start()...                                             //runs when the script is initialized, finds spawn points, sets base vals and starts spawn routine
Unity Message | 0 references
private void Update()...                                    //runs on tick, just checks the time of day to see if it needs to run start/end of day functions
0 references
void Spawn()...                                             //checks if spawn should be run, calls graphfun and runspawn
1 reference
public int GraphFun(float x)...                             //pushes the ratings through the quadratic function that calculates how many visitors should spawn
1 reference
public void Despawn(VisitorNpcAi NPC)...                    //runs to despawn the visitors and reduce current visitors variable
1 reference
void AvgVendorNum()...                                      //gets the average number of vendors/services in the scene
1 reference
public void RunSpawn(int NoVis)...                          //takes int a param, runs spawns iteratively relating to the int
1 reference
public void adjustSatisfaction(float individualSatisfaction)...  //modifies the overall satisfaction based on the individual visitors
1 reference
public void startOfDay()...                                 //runs at start of game day, resets starting money/total visitors, runs resetAllVendors and progessionRating
1 reference
public void resetAllVendors()...                            //runs the reset functions for all vendors reseting their daily visitor count
1 reference
public void dayRatingCalculation()...                       //runs functions and compiles day rating, then resets satisfaction and runs playerStatsEndOfDay
1 reference
public void resetSatisfaction()...                          //resets overall satisfaction
1 reference
public void EconRating()...                                 //calculates total profit and compiles an econ rating that will be combined to make the day rating
1 reference
public void VendorRating()...                               //creates a vendor rating based on range of visitors to each vendor, used in day rating
1 reference
public void TrackRating()...                                //creates a rating based on track quality score, used in day rating
2 references
public void getTrackQuality()...                            //used to get the total track score in scene
1 reference
public void getDecorations()...                             //gets all decorations in scene
1 reference
public void ProgressionRating()...                          //uses a number of factors to define how many visitors the park could take
1 reference
public void PlayerStatsEndOfDay()...                        //converts the day rating to a legible star rating and calculates profit for stats given to the player at the end of day
```

Overview of spawn system:

Day rating:

```
public void dayRatingCalculation()
{
    EconRating();
    VendorRating();
    TrackRating();
    //multiplies all subratings together
    dayRating = econRating*vendorRating*overallSatisfaction*trackRating;
    resetSatisfaction();
    PlayerStatsEndOfDay();
```

Day rating is calculated by multiplying the sub ratings together to get a single number that adjusts the spawn number.



Progression rating:

```
public void ProgressionRating()
{
    //gets required information
    getTrackQuality();
    AvgVendorNum();
    getDecorations();
    //bases rating on track score, number of vendors, number of tracks and number of decorations
    if(totTrackScore > 120 && avgNumVendors > 7 && numTracks > 6 && numDeco > 21)
    {
        progRating = 0.4f;
    }
    else if(totTrackScore > 80 && avgNumVendors > 5 && numTracks > 3 && numDeco > 14)
    {
        progRating = 0.3f;
    }
    else if(totTrackScore > 40 && avgNumVendors > 3 && numTracks > 1 && numDeco > 7)
    {
        progRating = 0.2f;
    }
    else
    {
        progRating = 0.1f;
    }
```

Progression rating is a much bigger change than the day rating, the overall idea is that the day rating is a minor adjustment on a daily scale, while the progression rating is a more long term rating based on the capacity of the park.

NpcSpawnerGraph

Graph in link above represents the spawn numbers “a” is the day rating and “b” is the progression rating.

The park opening and closing times are set in spawn:

```
void Spawn()
{
    //gets the current time of day
    float x = time.DayProgress;

    //this determines the park opening times, constants from the handle day night cycle sript
    if (time._isPast6PM == true || x < HandleDayNightCycle.PARK_OPENING_TIME)
    {
        return;
    }

    //multiplies by 10 to match equation
    x = x * 10;

    //runs graph fun using x as a param
    int a = GraphFun(x);

    //runs run spawn with results
    RunSpawn(a);
```

However to change them, they need to be changed in the HandleDayNightCycle script.

```
[Tooltip("Closes at 8PM (60 Mins * 20 Hours)")]
private const int LITERAL_PARK_CLOSING_TIME = 1200;

[Tooltip("Total minutes in a day.")]
private const int LITERAL_LENGTH_OF_DAY = 1440;


[Tooltip("0-1 value equivalent to 10AM")]
public const float PARK_OPENING_TIME = (float)LITERAL_PARK_OPENING_TIME / LITERAL_LENGTH_OF_DAY;

[Tooltip("0-1 value equivalent to 8PM")]
public const float PARK_CLOSING_TIME = (float)LITERAL_PARK_CLOSING_TIME / LITERAL_LENGTH_OF_DAY;

[Tooltip("Event that fires once at 9AM")]
public static event Action On_9AMReached;

[Tooltip("True after 9AM")]
private bool _isPast9AM = false;

[Tooltip("Event that fires once at 6PM")]
public static event Action On_6PMReached;

[Tooltip("True after 6PM")]
public bool _isPast6PM = false;
```

Increase how often visitors spawn:

```
void Start()
{
    //gets all spawn locations into array
    spawnPoints = GameObject.FindGameObjectsWithTag("SpawnPoint");
    //sets wait
    progRating = 0.1f;
    dayRating = 1f;

    //invoke repeating: after x seconds do function, repeat every x seconds
    InvokeRepeating("Spawn", 5f, 10.11f); //we use .11 to offset the 1 second animation so they don't look like a big marching band
```

In the start function, “Spawn” is invoked repeatedly , the first value is how long it waits to perform the first run of the function, the second is how often after that it runs the function, increasing this will make npcs spawn faster, careful with it.

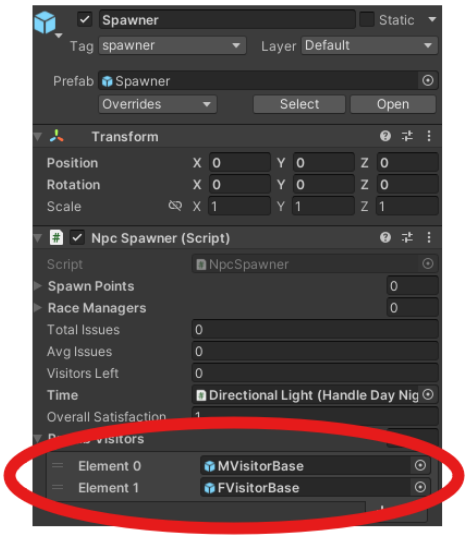
To add more visitor prefabs:

runSpawn:

```
public void RunSpawn(int NoVis)
{
    for(int i = 0; i < NoVis; i++)
    {
        // get random spawn location from list
        int location = UnityEngine.Random.Range(0, spawnPoints.Length);
        //sets location as a vector3
        Vector3 spawnLocation = spawnPoints[location].transform.position;
        //instanitates visitors using random prefab from list
        int x = rand.Next(0,prefabVisitors.Count);
        GameObject npc = Instantiate(prefabVisitors[x], spawnLocation, spawnPoints[location].transform.rotation);
        //increments variables
        VisitorsIn += 1;
        totalDayVisitors += 1;

        //gives the visiotr a ref to the spawner
        npc.transform.GetComponent<VisitorNpcAi>().spawner = this;
    }
```

Uses a list of gameobjects to spawn a random prefab, to add more to the list, find the spawner in scene and add it to the list there.



Economy rating:

Calculated by comparing the money at the start and end of the day, not really anything to change.

```
public void EconRating()
{
    //gets end of day money
    EndMoney = PlayerManager.Instance.Money;
    //calculated decimal profit in inverse so the lower the number the better
    econRating = startingMoney / EndMoney;
```

Vendor rating:

Calculated based on the range of number of visitors each vendor served each day, the lower the range the better as it means all vendors are valued around the same:

```
public void VendorRating()
{
    //list of ints containing how many visitors used each vendor/service
    List<int> vendorsVis = new List<int>();

    //gets the visitor numbers for all vendors/services
    GameObject[] x = GameObject.FindGameObjectsWithTag("Food Vendor");
    for (int i = 0; i < x.Length; i++)...
    x = GameObject.FindGameObjectsWithTag("Drink Vendor");
    for (int i = 0; i < x.Length; i++)...
    x = GameObject.FindGameObjectsWithTag("E Vendor");
    for (int i = 0; i < x.Length; i++)...
    x = GameObject.FindGameObjectsWithTag("Toilet");
    for (int i = 0; i < x.Length; i++)...

    //gets the higest and lowest visited vendors
    int highest = vendorsVis[0];
    for (int i = 0; i < vendorsVis.Count; i++)...

    int lowest = vendorsVis[0];
    for (int i = 0; i < vendorsVis.Count; i++)...

    //rating depends on the range of visitors at vendors
    if (highest - lowest > 30)
    {
        vendorRating = 1.3f;
    }
    else if (highest - lowest > 20 && highest - lowest < 30)
    {
        vendorRating = 1.1f;
    }
    else if (highest - lowest > 10 && highest - lowest < 20)
    {
        vendorRating = 0.9f;
    }
    else if (highest - lowest > 5 && highest - lowest < 10)
    {
        vendorRating = 0.7f;
    }
```

Can change the bottom of the function, adjusting the weighting of the rating or conditions for each level, can change the floating rating values or the flat conditional ones.

Track rating:

```
public void TrackRating()
{
    //gets the track quality
    getTrackQuality();
    //compares to previous day to see if the track has improved
    if(totTrackScore > prevTrackScore)
    {
        prevTrackScore = totTrackScore;
        trackRating = 0.8f;
    }
    else if(totTrackScore == prevTrackScore)
    {
        prevTrackScore = totTrackScore;
        trackRating = 1f;
    }
    else
    {
        prevTrackScore = totTrackScore;
        trackRating = 1.2f;
    }
```

The track rating is based on whether the track quality score has improved each day, going up if it has down if its gotten worse and remaining 1 if it hasn't changed, can change these values, if you want to punish players for not improving instead of the score remaining still, i.e. if the score hasn't changed, the rating is negative.

Progression Rating:

```
public void ProgressionRating()
{
    //gets required information
    getTrackQuality();
    AvgVendorNum();
    getDecorations();
    //bases rating on track score, number of vendors, number of tracks and number of decorations
    if(totTrackScore > 120 && avgNumVendors > 7 && numTracks > 6 && numDeco > 21)
    {
        progRating = 0.4f;
    }
    else if(totTrackScore > 80 && avgNumVendors > 5 && numTracks > 3 && numDeco > 14)
    {
        progRating = 0.3f;
    }
    else if(totTrackScore > 40 && avgNumVendors > 3 && numTracks > 1 && numDeco > 7)
    {
        progRating = 0.2f;
    }
    else
    {
        progRating = 0.1f;
    }
```

Uses a number of factors the change the progression rating, the progression rating drastically changes the spawn rate of visitors even with just 0.1 changing at a time, so large conditions are recommended, but all number here can be played with and are fairly easy to understand.