Involved Classes:

* WaveArray: only contains hardcoded values for the 2D array of all waves. Each row in the array is formatted as:  
  [TimeToFinish, SpawnTimer, T1, T2, T3, T4, S1, S2, S3, S4, H1, H2, H3, H4]
* TimerSystem: Attached to Timer Display. Used to obtain the elapsed time to control waves, using the public integer shortElapsedTime
* CartesianAndPolar: Converts between cartesian and polar coordinates to resolve spawnpoints
* Enemy: To instantiate the appropriate enemy, and set its health

**EnemySpawner class**

// The following are set in Inspector:  
create float spawnMultiplier   
create speedMultiplier  
create float spawnRadius   
create float freeplayRampupMultiplier   
create GameObject timerObject  
create int enemyHealthTypes  
create GameObject list enemyList

create integer waveNumber   
create integer maxWaveNumber   
create float spawnCooldown  
create float array currentWave  
create TimerSystem gameTimer

Start procedure:

set spawnCooldown to 0  
set waveNumber to 1  
set maxWaveNumber to length of dimension 0 of WaveArray’s waveArray array  
set currentWave to WaveArray.waveArray’s 0th index  
set gameTimer timerObject’s TimerSystem component

Update procedure:

if (gameTimer.shortElapsedTime >= currentWave[0] AND waveNumber < maxWaveNumber):  
 set waveStartTime to currentWave[0]  
 set currentWave to WaveArray.waveArray[waveNumber]  
 add 1 to wavenumber  
else if (wavenumber == maxWaveNumber):  
 add (Time.deltaTime \* freeplayRampupMultiplier) to spawnMultiplier

if (spawnCooldown >= currentWave[1] \* spawnMultiplier):  
 Execute SpawnEnemy()  
 set spawnCooldown to 0

add Time.deltaTime to spawnCooldown

SpawnEnemy function:

create float spawnAngle and set it to a random number between 0 and 360  
create int randomEnemy  
create bool isEnemyChosen and set it to false

create int enemyHealth  
create int enemyType  
create float array enemyCoOrds

while (isEnemyChosen = false):  
 set randomEnemy to a random integer between 2 and length of currentWave   
 if (currentWave[randomEnemy] != 0)  
 set isEnemyChosen to true

subtract from randomEnemy by 2

set enemyHealth to 1 plus (mod of randomEnemy by enemyHealthTypes)  
set enemyType to floor of randomEnemy divided by enemyHealthTypes  
set enemyCoOrds to CartesianAndPolar’s function ConvertToCartesian(spawnRadius, spawnAngle)

create GameObject newEnemy and Instantiate it as enemyList[enemyTypeID] at position: (enemyCoOrds[0], enemyCoOrds[1])  
rotate newEnemy by spawnAngle + 90 degrees  
in newEnemy's Enemy component, execute CreateEnemySettings(enemyHealth)