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**

create float spawnMultiplier and set it to 1

create float spawnRadius and set it to 16

create float freeplayRampupMultiplier and set it to 0.01

create integer waveNumber

create integer maxWaveNumber

create float spawnCooldown

create float array currentWave

create TimerSystem gameTimer

create public gameObject enemyTriangle  
create public gameObject enemySquare  
create public gameObject enemyHexagon

Start function:

set spawnCooldown to 0  
set waveNumber to 1  
set maxWaveNumber to length of dimension 0 of WaveArray.waveArray  
set currentWave to WaveArray.waveArray[0]  
find Timer Display, and set gameTimer to the TimerSystem component

Update function:

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 4)  
set enemyType to 1 plus (floor of randomEnemy divided by 4)  
set enemyCoOrds to ConvertToCartesian(spawnRadius, spawnAngle)

Look at cases for enemyType:  
case 1:  
 create GameObject newEnemy and Instantiate it as enemyTriangle at position: (enemyCoOrds[0], enemyCoOrds[1])  
case 2:  
 create GameObject newEnemy and Instantiate it as enemySquare at position: (enemyCoOrds[0], enemyCoOrds[1])  
case 3:  
 create GameObject newEnemy and Instantiate it as enemyHexagon at position: (enemyCoOrds[0], enemyCoOrds[1])

rotate newEnemy by spawnAngle + 90 degrees  
in newEnemy's Enemy component, execute CreateEnemySettings(enemyType, enemyHealth)