Base Gameplay Concepts:

- A 2D shooter
- Player against bots
- To increase level (health, damage, speed, etc.) the player has to defeat enemies with higher level stats
    - Once a stronger enemy is defeated, the player has the chance to “capture” the enemy’s stats in the form of a “star” object
        - Star object has an expiration date. If players take too long to grab the star, it will vanish
        - If the star object is collected, the enemy respawns with the player’s previous stats; if not collected, the enemy respawns with the same stats. Both events occur after a cooldown
    - Doing so will heal all player health
- If the player dies to an enemy, they will also leave behind a star object
    - Any enemy colliding with the star object will collect the star. That enemy will gain the player’s previous stats; the player will respawn with that enemy’s old stats
        - Stats switch
        - Higher level enemies will avoid the star objects left behind by lower

players/enemies but can possibly be tricked or forced into picking them up
        - Blocking an enemy in a corner, with the only way out forcing them to collect the star
    - This mechanic encourages players to be cautious to whom they die, and where
        - A player with low health may try to die in an area filled with equal or higher level enemies to increase their chances of “trading up” rather than down
- Players and enemies do not heal over time
    - Healing only occurs through respawning or gaining new stats, forcing player mobility
- If the player dies at the base stat level, the game is over
- The game is won by being the first to defeat a final boss
- Environment may include basic obstacles (impassable tiles, structures) and powerups
    - Powerups might temporarily increase shooting speed, movement speed, or damage

Implementation Tools:

- Unity Engine 2019
- Visual Studio
- C#

Assets:

- Player sprite
- Enemy sprite
- Bullet sprite
- Star sprite
- Environments (background images, obstacles, tilemaps)
- Powerups (temporary boosts to damage, speed, health)
- C# scripts:
    - PlayerHandler
        - Manages player stats, player shooting mechanics, player movement, player death
    - EnemyHandler
        - Manages enemy stats, enemy movement, enemy death, enemy shooting, star stat setting
    - EnemySpawner
        - Manages spawning enemies; start up spawn and continuous spawn
        - Selects enemy levels based on player level and number of enemies on the screen
    - StarHandler
        - Sets enemy stats to a player once picked up; causes stars to expire if not picked up
    - Projectile

        - Dictates lifespan of projectiles, determines what happens during projectile collisions, removes health from player/enemies
    - PowerupHandler
        - Transfers powerups to players on collection, expires old powerups
    - PowerupSpawner
        - Spawns random powerups based on timing + number on map
    - GameHandler
        - Tracks score, game difficulty, pause/start, end game, UI

Possible Additional Concepts:

- Online multiplayer
    - Game would feature a number of bots depending on the number of players
    - Players could play every man for himself, or as teams
- Instead of a bullet hell game, make it a 2D platformer/roguelike
    - Allows for more fun with graphics, animations, as well as different weapons
    - Sounds a bit like Hades

Scripts To-Do:

- Player respawn
    - After a timer, to allow for enemies to collide with dropped star
    - Exception: player is already at lowest level
- Player drop star
    - Send stats to star
    - Update StarHandler to set stats to enemy
- Stats by level
    - A default set of stats per level
    - Update EnemySpawner to generate stats based on level
- Spawn continuously*
    - Update EnemySpawner to create new enemies based on:
        - Player level
        - Number of enemies on screen
        - Levels of enemies on screen
- Spawn enemies off screen only
- Add UI
    - Score
    - Main menu
        - How to play
        - Credits
    - Pause/exit
    -

Spawn Continuous Enemies Sample Conditions*:

- Conditions increment by 3

    - Levels 1-3, 4-6, 7-9, etc.
- Number of enemies total is 2x player level with a base of 6
    - Ex. levels 1-3, 6 enemies total. level 4, 8 enemies total
    - Maintain these levels throughout the game; if an enemy dies, replace it with another enemy of randomly generated level pending level range
        - Ex. if player level is 5, replacement enemy level may range 1-6
- Number of highest level enemy to spawn at each range increases by 1
    - Ex. levels 1-3, one level 3 enemy at a time. levels 4-6, two level 6 enemies at a time
    - If there is more than the max number of max level enemies at a time, do not spawn more
        - Ex. level range 4-6, with two level 6 enemies spawned, do not spawn more level 6 enemies
- All other enemies generate with random levels, pending level range. Percent likelihood for each range
    - Ex. default game start, one level 3 enemy, five enemies with random levels 1-2
    - Ex. level range 4-6, enemies level 1-2 might have 45% of spawning, 3-4 35%, 5 20% **

* single player concept

** would need a more consistent calculation for this. Idea being lower level enemies are more likely to be spawned, but as the player’s level range increases, the variety of enemy levels also increases, and the percentages should change accordingly

These would perhaps be starting conditions for playtesting; playtesting would be required to determine how the game should account for possible rises in enemy level.