First day:

For first day goal is to set up space shooter with scrolling background and enemies to shoot at.

Folder structure is as follows:



* Assets folder containg all game art.
* Scripts folder containg external scripts, in this case Phaser engine script.
* Src folder contains our game code.
* Index.html is start page.

Steps:

1. Student, in game.js file property for game width is already defined. Please define property for height. Set value to 600.
2. In game.js file add update function to mainPhaserFunctions object.
3. In BaseGameObject file, add sprite property.
4. In BaseGameObject file, add update function to prototype of BaseGameObject
5. In ScrollingBackground file, call base game object and pass required arguments
6. In BaseGameObject file, pass sprite variable to enable physics for ScrollingBackground. Use this keyword.
7. In game.js file , initialize backgrounds in create method to array.
8. In game.js , push another scrolling background to backgrounds array. Sprite for that background should have -WIDTH value for x position and 0 for y position.
9. In game.js file, update method, create loop for backgrounds array and call update method foreach game entity.
10. In ScrollingBackground update method check if sprite position is less then or equal negative width. If so, set it to window width. After this step try to run game and see if background is scrolling.

Pause – Explain background scrolling

1. In Player file, add parameters to Player constructor
2. In Player file , inherit from BaseGameObject
3. In Player file, create player prototype.
4. Player file, make sprite body collidable with world bounds

15. Player file: in update method add check for left and right key. Also set appropriate speed to velocity.

16. In game.js file, create player instance.

Try to see if player is moving

17. In BaseEnemy.js, check if sprite has passed left side of screen. If so, set active property to false. Take sprite width into account

18. In Asteroids.js file, Create Asteroid class, add appropriate parameters. Inherit from BaseEnemy. Set negative velocity and random rotation for Asteroid.

19. In game.js file, create property enemies and initialize it as array type.

20. Loop through enemies array. Enemies array should loop from end of array to begining of array.

21. Update enemies. If enemies active property is set to false, remove enemies from array. Use array splice method to do so.

22. Uncomment asteroid spawn method and asteroidSpawnTimer

23. Create random int ( whole number) between 0 and 7 ( 0 and 7 included) and use it to create asteorid sprite. Use parseInt and Math.random functions. See asteorid sprite naming in preload method.

Create random y position for sprite that goes from yOffset to HEIGHT - yOffset. X position should be set to WIDTH + xOffset.

24. Create asteorid, pass in this and sprite variable. Add asteroid to enemies array.

25. Uncomment collision code and add index

26. In BaseGameObject.js file add object property to sprite, and set it to this.

27. In Player.js file add health property and set it to 100.

28. In Asteroid.js file, add damage property and set it to some number, like 20, 25 ...

29. In game.js file , Decrease player health on collision with sprite. Use damage property from asteroid. Set enemy active property to false.

30. If player health is less then 0, set active property to false. Destroy player sprite.

Try running game after this step. Enemies should dissapear after collision.

31. In BaseBullet.js file , create BaseBullet constructor and set appropriate properties.

32. In RegularBullet.js file , call base bullet update method

33. In player.js file add bind key method to player prototype, this way we can add new keys to player. We need this method to add shooting key.

34. Add check key down method, pass key as parameter

35. Add get position method to player, which will return sprite position.

36. In game.js file create spawnRegularBullet method, which will just add new RegularBullet. Use player position for bullet sprite position and set bullet direction

37. Bind shooting key to player, like space

38. Check if shooting key is pressed. If so, call spawnRegularBullet

Try running game now to see bullets.

39. Add COOLDOWN property to window object. Next to WIDTH and HEIGHT

40. Add cooldown property and set it to COOLDOWN

41. If game is run now bullets are spawned really fast, let's use cooldown property and reduce it by

this.time.physicsElapsed ref: <http://phaser.io/docs/2.6.1/Phaser.Time.html#physicsElapsed>

42. Add check for cooldown <= 0 and reset this.cooldown to COOLDOWN inside if statement.

43. Add for loop for bullets starting from bullets end, update bullet,

add check for inactive bullets, if bullets is not active, remove it from array and destroy bullet sprite

44. Add for loop for bullets and check for collision between bullet and enemy. Pass bulletEnemyCollision to collide call

45. Set sprite objects active property to false inside bulletEnemyCollision function.

46. Finally add offset to in bullets spawn function to make bullets spawn at correct point.