**Game Manager**

Params:

* healthThreshold
* maxHealth
* healthDecay
* addedHealth

State:

* instance
* currentTime
* checkpointTime
* checkpointsTime
* cpTimeIdx
* currentHealth
* totalTime
* lastCollidedObject

Properties:

* Instance
* uIManager
* lastCheckpoint
* player
* carSmoke
* carExplosion

Update:

* Starts Analytics and sets the max health and decay rate to specified variables.
* Awake()
* LastCheckPoint()
* PlayerCollision()
* Gameover()

Events:

* Displays stopwatch and time since entering the last checkpoint. Also gets current player health and if it is below the specified threshold, smoke forms on the car.
* Destroys the second instance of the GameManager (instance).
* Records the current checkpoint and converts time to a string in order to push the current checkpoint time into an array. It also sends the current player health to the analytics, adds health to the players health bar and restarts the stopwatch for the following checkpoint.
* Calls the HealthBarManager to reduce health and records last obstacle hit by the player (car).
* If the player has failed to reach all checkpoints, the car explosion is set to True and activated.
* The game over panel is shown and the game is paused.
* The players (car) “death information” is sent to the analytics, which contains; the total time, players position and the last object the player had collided with.

**UI Manager**

Params:

* stopwatchText
* gameoverPanel
* resultText
* cpText

State:

* currentTime

Update:

* UpdateTime()
* GameOverScreen()
* RestartScene()

Events:

* Displays the stopwatch which contains minutes, seconds and milliseconds.
* Activates game over panel and displays “YOU WIN!” if the player has reached all checkpoints and “YOU DIED!” if the player failed to reach all checkpoints.
* When the game ends, a panel with text is displayed at the center of the screen, containing the exact times the player reached each checkpoint.
* The game is restarted when the “Play Again” button is selected.

**Player Movement**

Params:

* forwardSpeed
* reverseSpeed
* turnSpeed
* speed
* downForce
* raySize

State:

* dx
* dy
* grounded

Properties:

* carBody
* car
* groundLayer
* rayPoint

Update:

* declare horizontal and vertical inputs.
* FixedUpdate()

Events:

* Car should move forward & back using ‘AddForce’ based on vertical Input.
* Car should rotate 30o either left & right based on horizontal Input.
* Only when the car is grounded will the player be able to steer (rotate).
* Forward force is applied to the player (car) when it is moving forward at a speed greater than 0.
* Downwards force is applied to the player (car) when it is moving.

Player

**Health Bar Manager**

Params:

* currentHealth
* healthDecay
* addedHealth
* instance
* Instance

State:

* healthSlider

Update:

* Awake()
* MaxHealth()
* SetHealthDecay()
* SetAddedHealth()
* SetHealth()
* AddHealth()

Events:

* Destroys gameObject (healthSlider) …
* Sets the max health of the player (car).
* Sets the health decay which effects the players health bar.
* Sets the amount of health added to the players health bar.
* Adjusts the health value of the players health bar when called.
* Adds a new health value to the players current health value.

**Player Health**

State:

* minCollisionDamage

Properties:

* minCollisionDamage

Update:

* OnCollisionEnter()

Events:

* Identifies any game objects the player (car) comes in contact with that contain the tag “Obstacle”.
* With a tunable collision radius, depending if the force is greater than the tunable public variable; the player loses health.

Activates

**Sends events to**

Affects

Activates

Sends events to

Follows

Checkpoint Prefab

**CheckpointManager**

Params:

* Instance

State:

* instance

Update:

* Initializes the current checkpoint and sets every checkpoint to “inactive” excluding the first one.
* Awake()
* NextCheckPoint()

Events:

* Move to the following checkpoint once the one before has been reached.
* Calls the game over function in the GameManager once the last checkpoint has been reached by the player (car).
* Destroys the second instance of the CheckpointManager (instance).
* Activates the following checkpoint after the one before has been reached.

Camera

**Camera Movement**

Params:

* offset
* decay

State:

* targetX
* targetY
* targetZ

Properties:

* target
* targetSubObject
* lookAtTarget

Update:

* FixedUpdate()

Events:

* Declares all target angles based on the players (car) transform angle.
* Declares the cameras position as a Vector of the targetSubobject and addition of the offset.
* Lerps through the cameras position with regard to the players position, camera position and decay.
* Identifies and follows all angles the player (car) moves.

**Checkpoint**

Params:

* activeColor
* inactiveColor
* radius

State:

* cpRenderer
* isActive
* sphereCollider

Update:

* sets the detection radius by resizing the spheres box collider.
* OnTriggerEnter()
* ChangeInactiveColor()
* ChangeActiveColor()

Events:

* Identifies when the player object (car) collides with an Active checkpoint, calls on the CheckpointManager and sets the colour to the Inactive state.
* When the player (car) has collided with a checkpoint it will change the colour of the checkpoint to “inactiveColor”.
* If the player has not yet collided with a checkpoint it will remain in the “activeColor” state.

Initializes

Activates