MOON MISSION

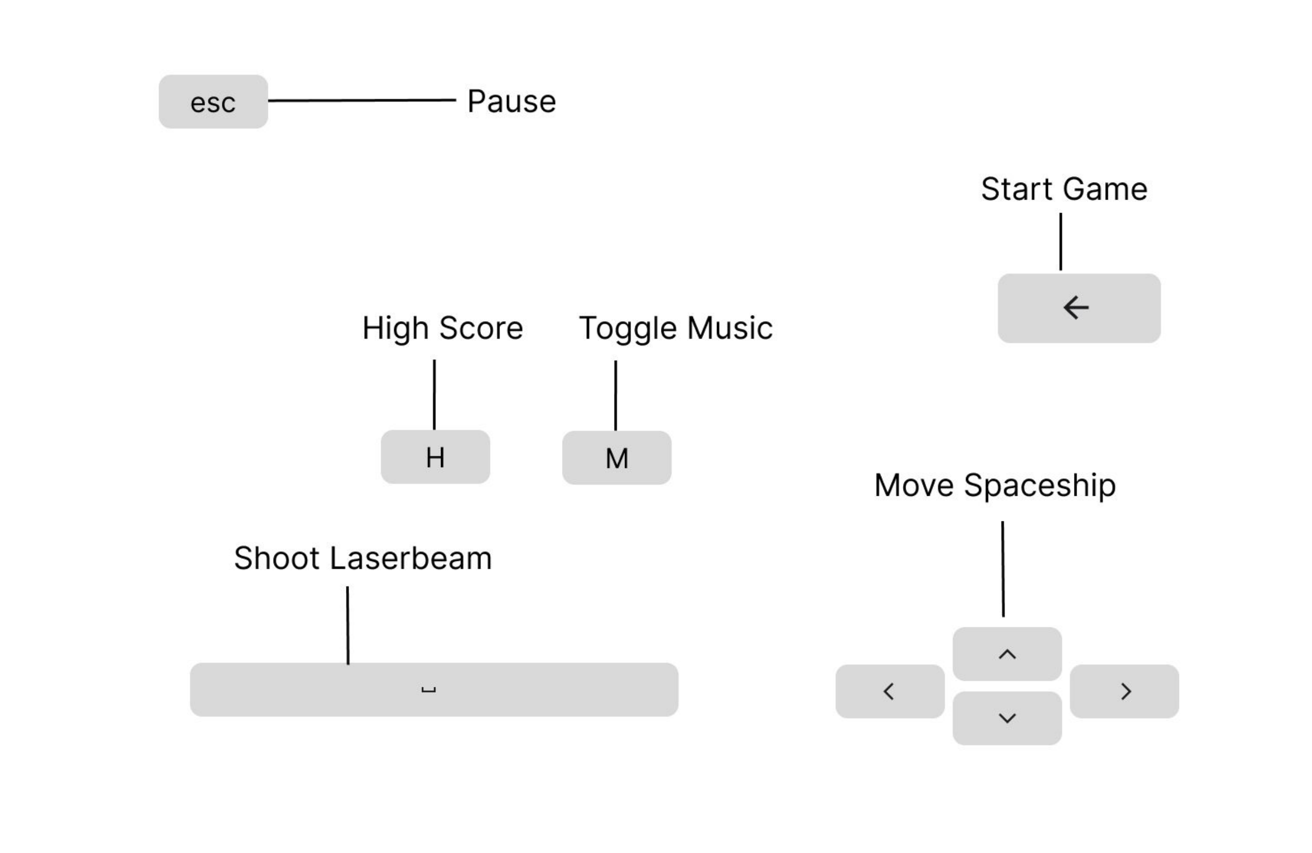
Parham Berenjian, Carl Hasselblad, Thomas Ingvarsson,
Caisa Köhlin, Madeleine Gustafsson, Yehad Moussaoui

Spelbeskrivning

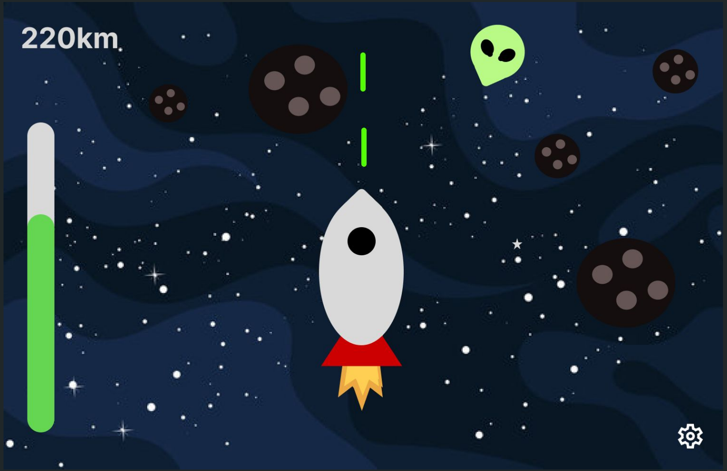
- Rymdraket
- Åk så långt du kan!
- Full syremätare
- Game over
    - Slut på syre
    - Kollision med fiender
- Laserbeam
- Speed Boost
- Score och High Score i Local Storage

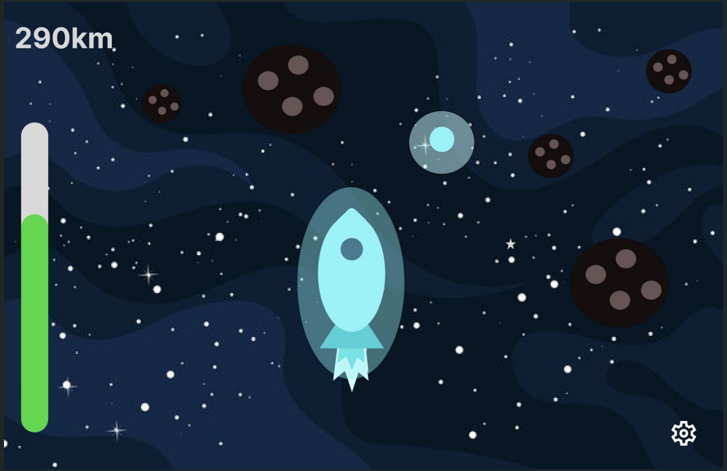
CK

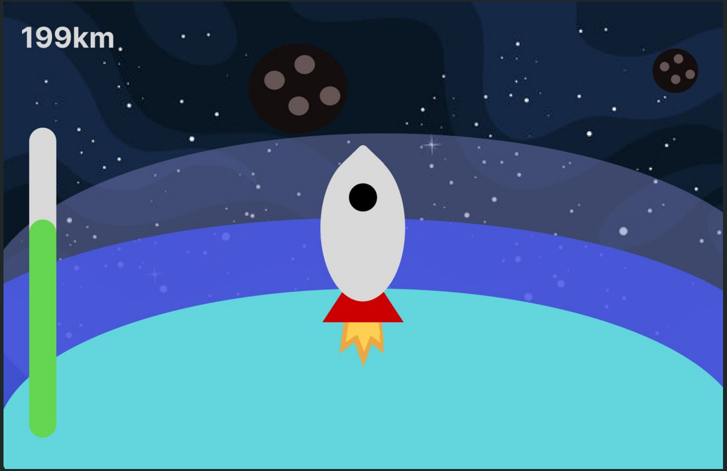
Interaktionsskiss - Slutgiltig version

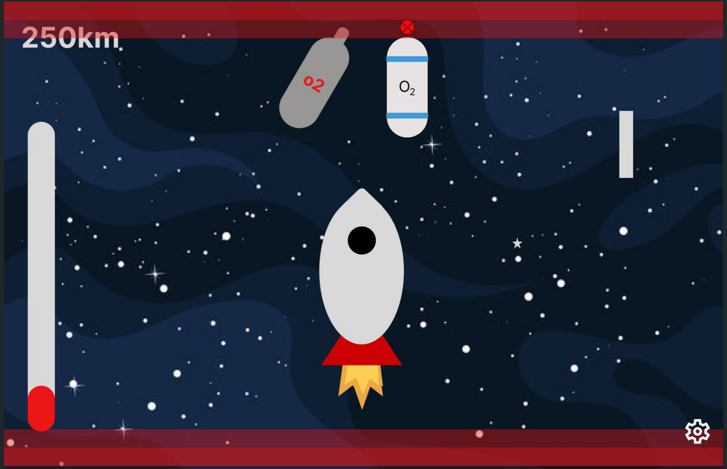


CK

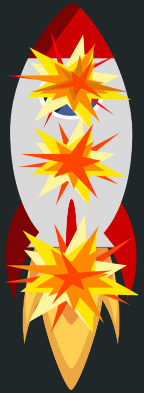

220km

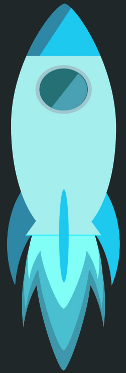

290km

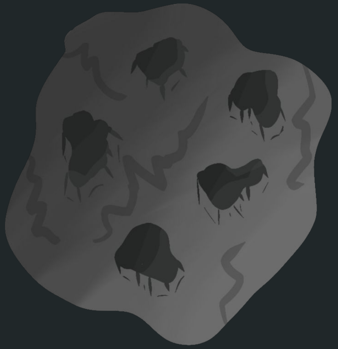

199km


250km
o2
O_2

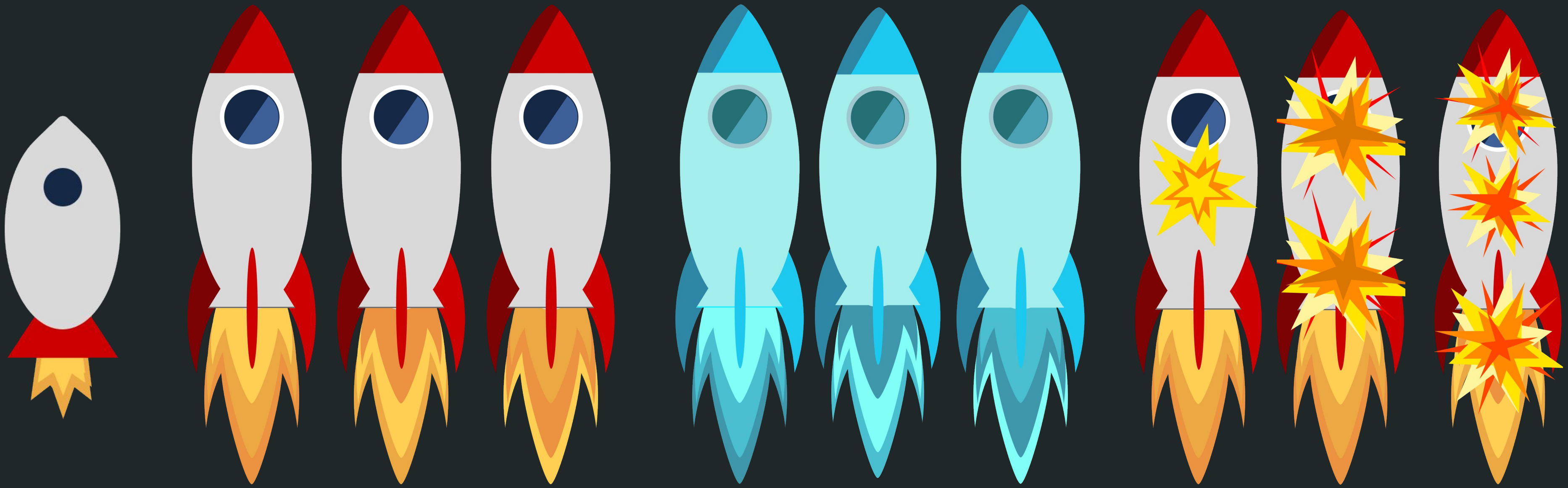
Nya features som tillkommit



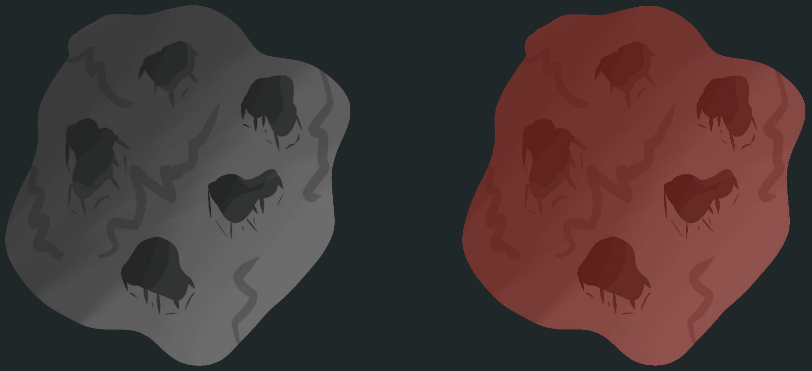


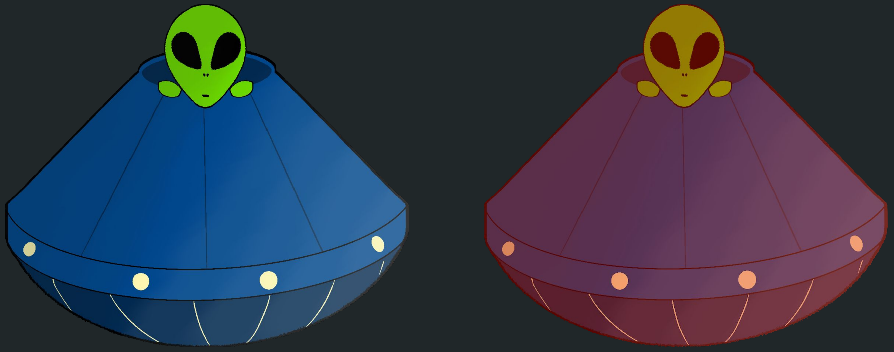


Game Entities - Spaceship



Game Entities





Asteroid

HP: 2

Spawn: 1-2 sekunder

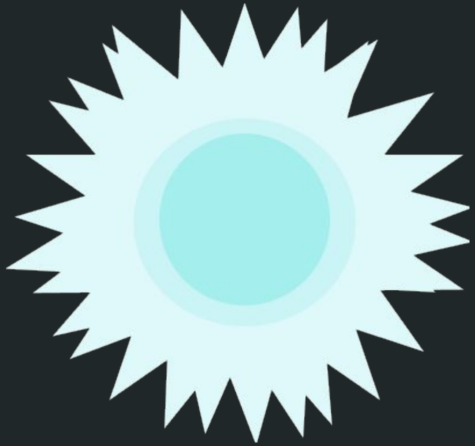
Alien

HP: 3

Spawn: 1-5 sekunder

CK

Game Entities



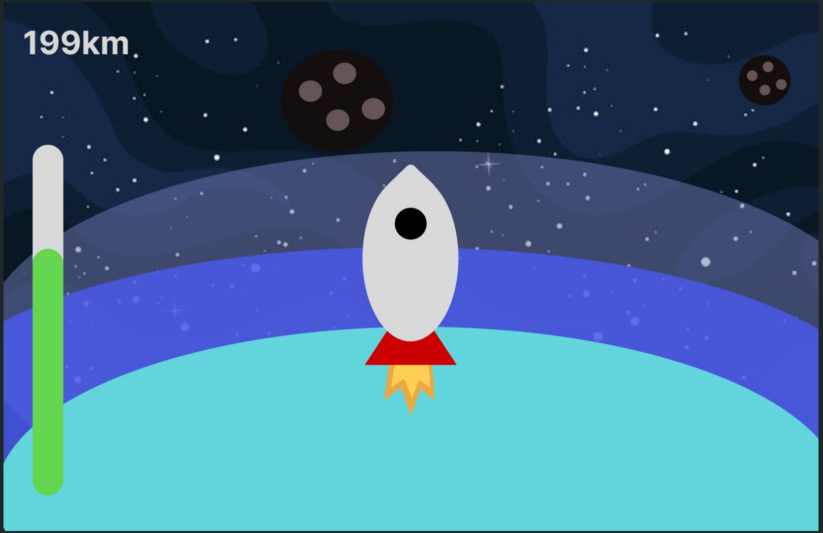
Oxygen Tank

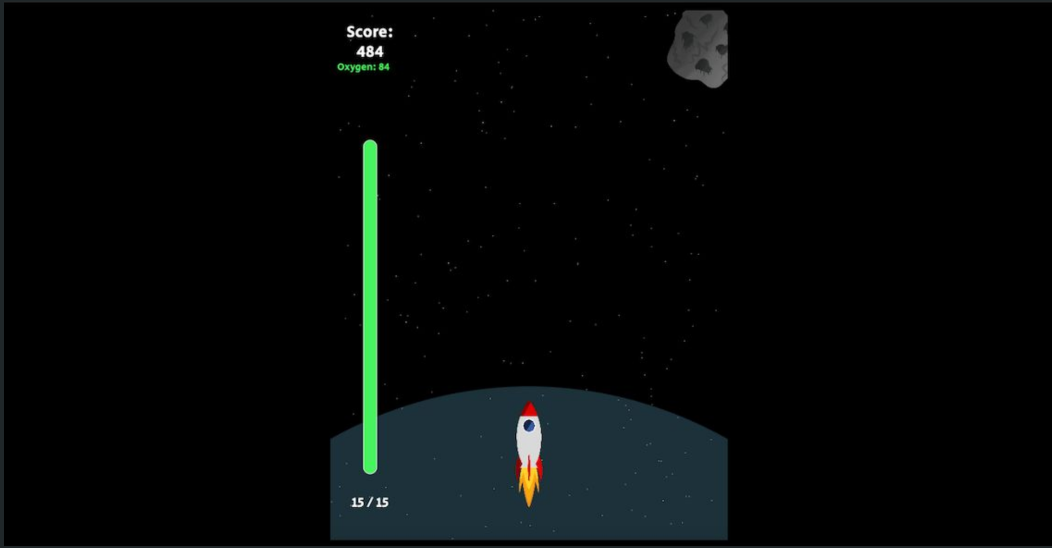
Spawn: 1-20 sekunder

Speed Boost

Spawn: 1-20 sekunder

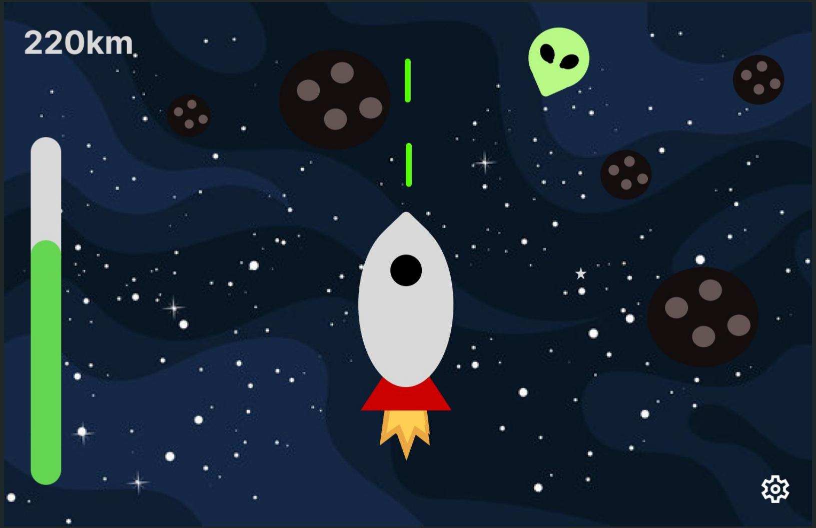
First Design | Gameplay

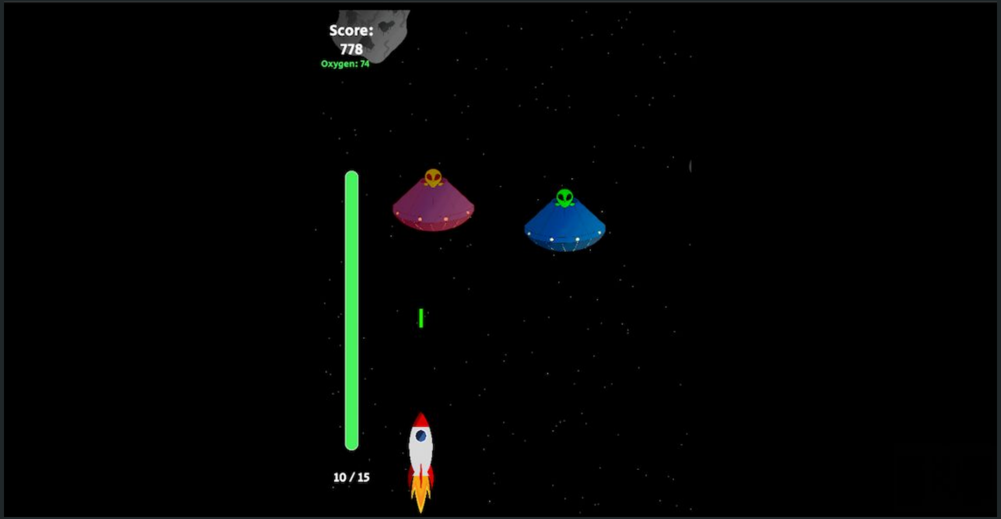




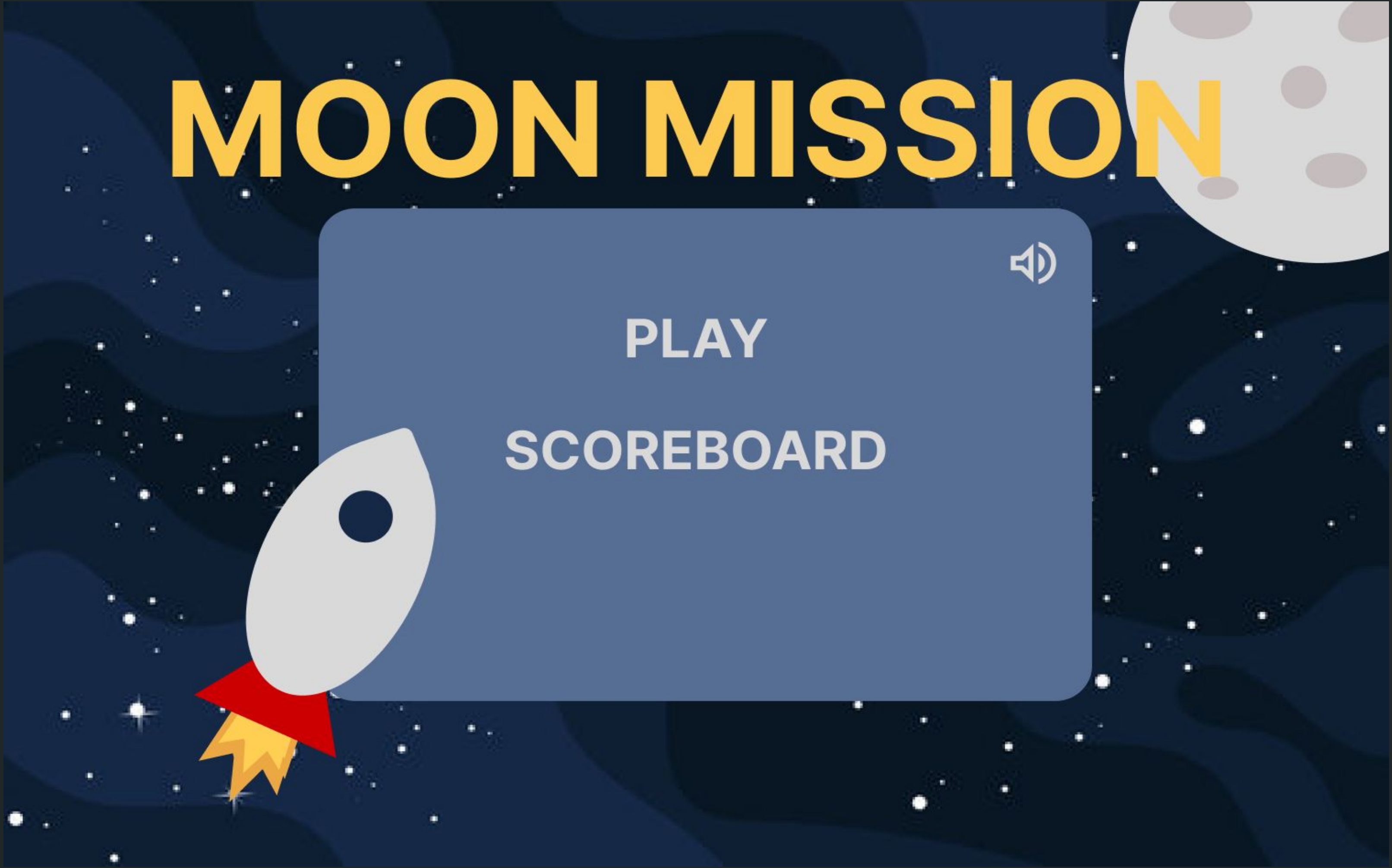
CK

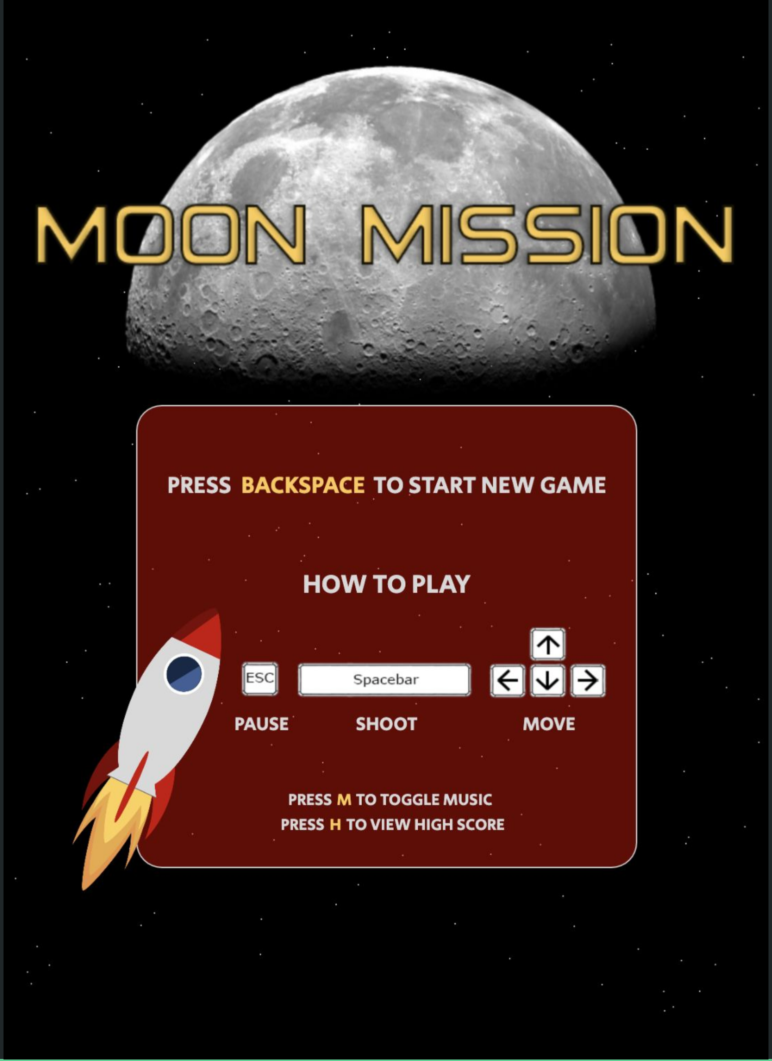
First Design | Gameplay





CK


MOON MISSION
PLAY
SCOREBOARD



MOON MISSION

PRESS BACKSPACE TO START NEW GAME

HOW TO PLAY

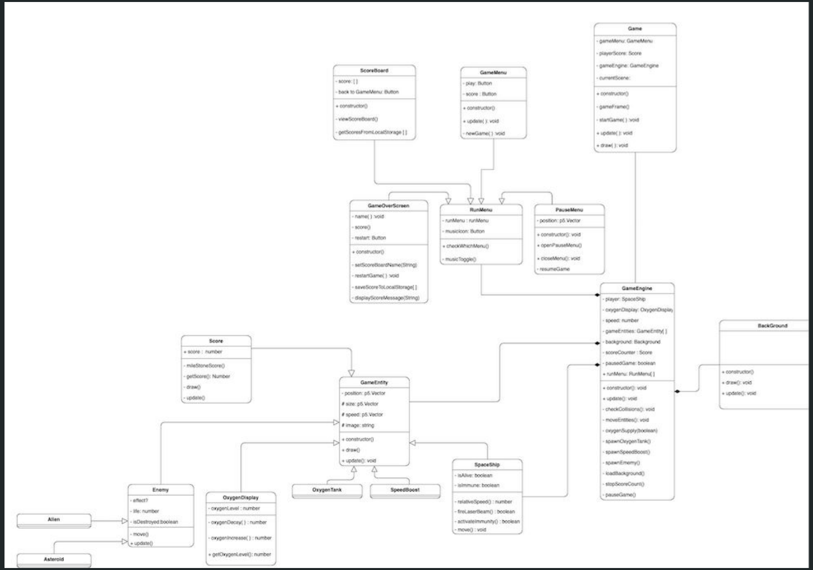
PRESS M TO TOGGLE MUSIC
PRESS H TO VIEW HIGH SCORE

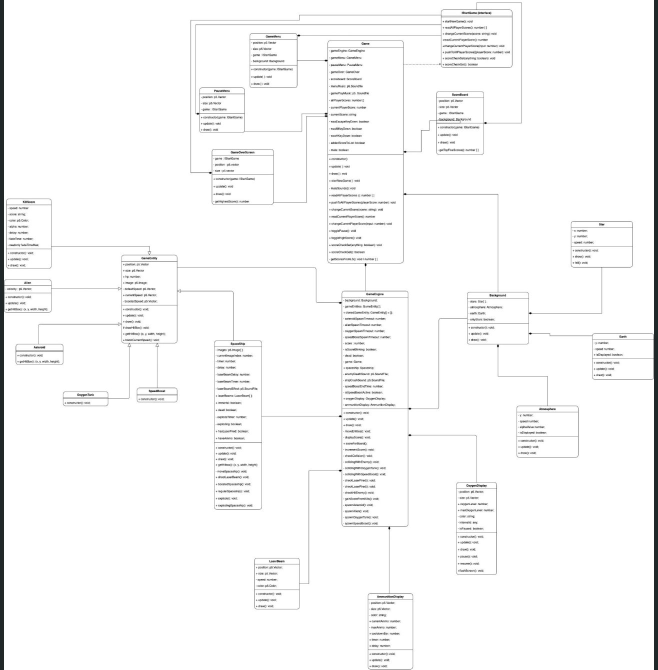
CK

Klasschema

Första version

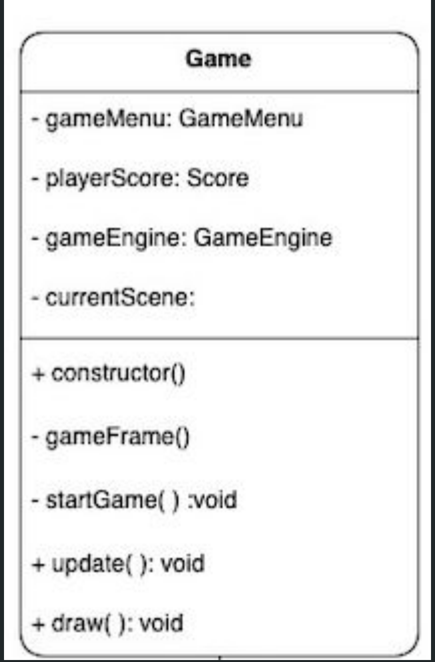
Slutgiltig version





CH

Första version



Game

- gameMenu: GameMenu

- playerScore: Score

- gameEngine: GameEngine

- currentScene:

+ constructor()

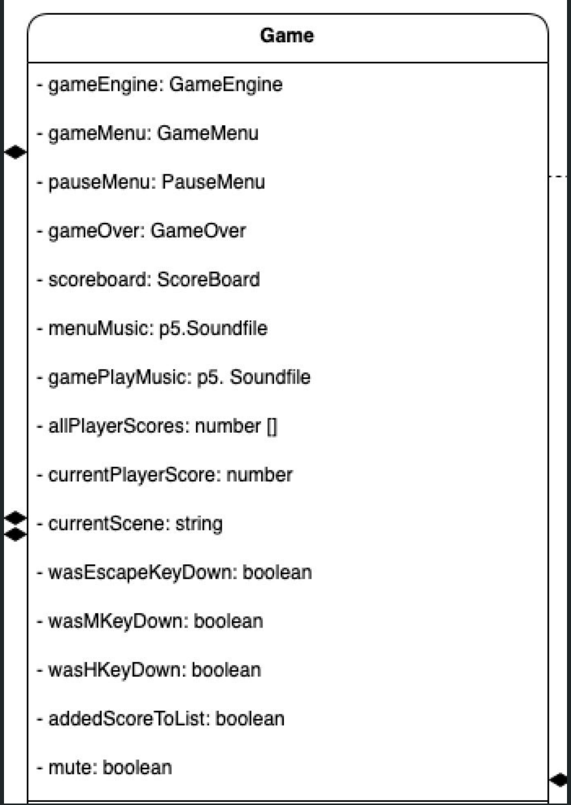
- gameFrame()

- startGame( ) :void

+ update( ): void

+ draw( ): void

Slutgiltig version



Game

- gameEngine: GameEngine

- gameMenu: GameMenu

- pauseMenu: PauseMenu

- gameOver: GameOver

- scoreboard: ScoreBoard

- menuMusic: p5.Soundfile

- gamePlayMusic: p5. Soundfile

- allPlayerScores: number []

- currentPlayerScore: number

- currentScene: string

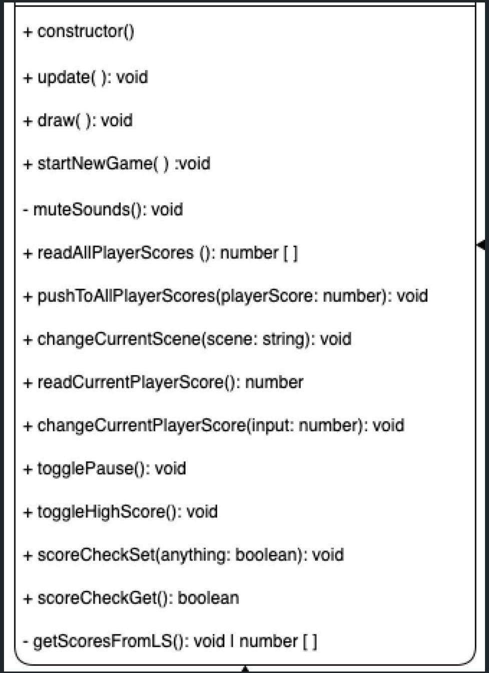
- wasEscapeKeyDown: boolean

- wasMKeyDown: boolean

- wasHKeyDown: boolean

- addedScoreToList: boolean

- mute: boolean



+ constructor()

+ update( ): void

+ draw( ): void

+ startNewGame( ) :void

- muteSounds(): void

+ readAllPlayerScores (): number [ ]

+ pushToAllPlayerScores(playerScore: number): void

+ changeCurrentScene(scene: string): void

+ readCurrentPlayerScore(): number

+ changeCurrentPlayerScore(input: number): void

+ togglePause(): void

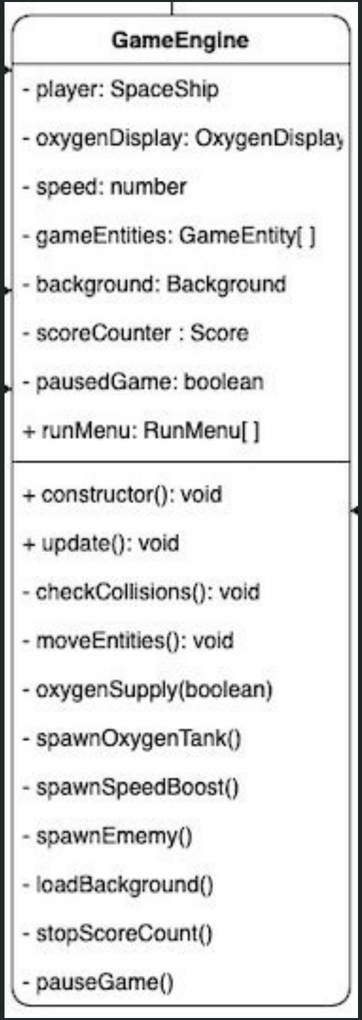
+ toggleHighScore(): void

+ scoreCheckSet(anything: boolean): void

+ scoreCheckGet(): boolean

- getScoresFromLS(): void I number [ ]

Första version

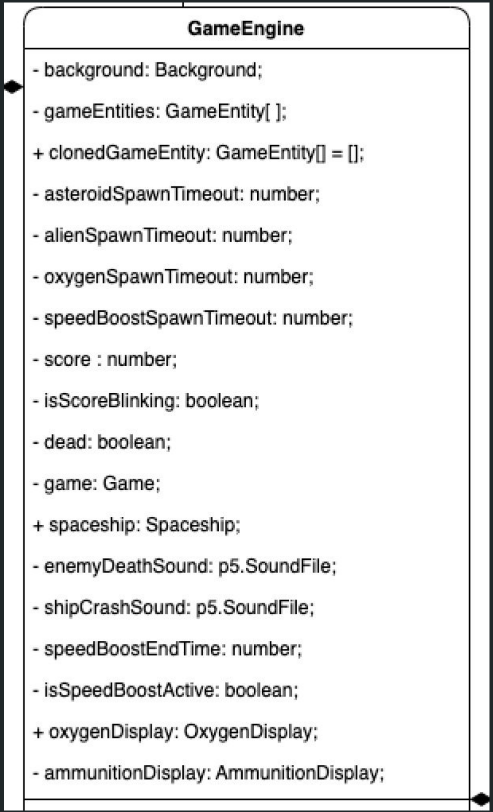


GameEngine

- player: SpaceShip
- oxygenDisplay: OxygenDisplay
- speed: number
- gameEntities: GameEntity[ ]
- background: Background
- scoreCounter : Score
- pausedGame: boolean
+ runMenu: RunMenu[ ]

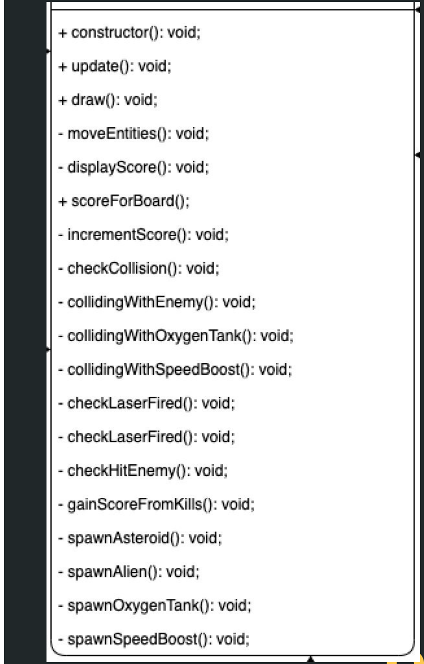
+ constructor(): void
+ update(): void
- checkCollisions(): void
- moveEntities(): void
- oxygenSupply(boolean)
- spawnOxygenTank()
- spawnSpeedBoost()
- spawnEmemy()
- loadBackground()
- stopScoreCount()
- pauseGame()

Slutgiltig version



GameEngine

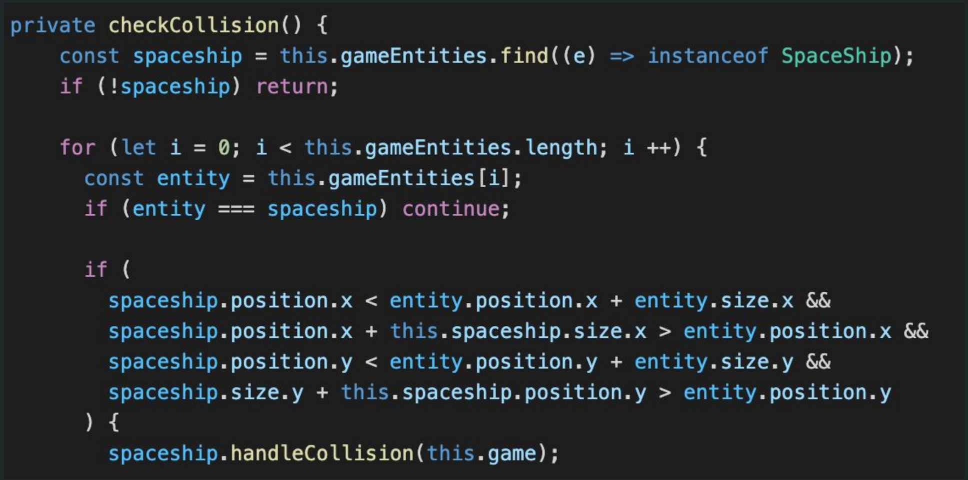
- background: Background;
- gameEntities: GameEntity[ ];
+ clonedGameEntity: GameEntity[] = [];
- asteroidSpawnTimeout: number;
- alienSpawnTimeout: number;
- oxygenSpawnTimeout: number;
- speedBoostSpawnTimeout: number;
- score : number;
- isScoreBlinking: boolean;
- dead: boolean;
- game: Game;
+ spaceship: Spaceship;
- enemyDeathSound: p5.SoundFile;
- shipCrashSound: p5.SoundFile;
- speedBoostEndTime: number;
- isSpeedBoostActive: boolean;
+ oxygenDisplay: OxygenDisplay;
- ammunitionDisplay: AmmunitionDisplay;



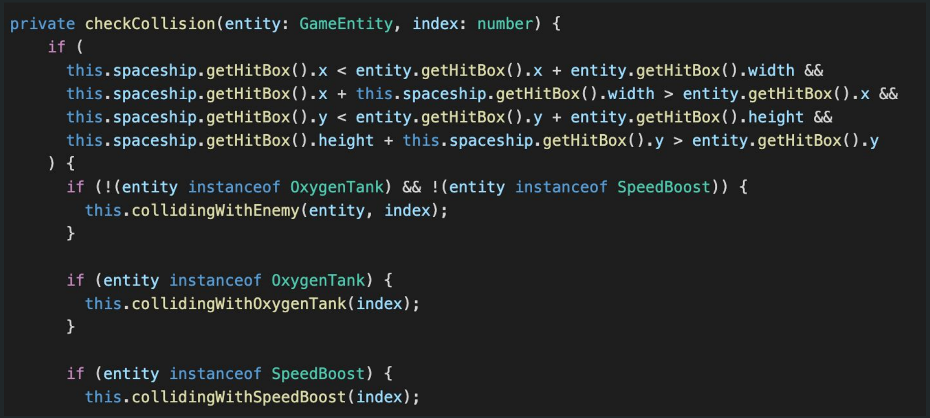
+ constructor(): void;
+ update(): void;
+ draw(): void;
- moveEntities(): void;
- displayScore(): void;
+ scoreForBoard();
- incrementScore(): void;
- checkCollision(): void;
- collidingWithEnemy(): void;
- collidingWithOxygenTank(): void;
- collidingWithSpeedBoost(): void;
- checkLaserFired(): void;
- checkLaserFired(): void;
- checkHitEnemy(): void;
- gainScoreFromKills(): void;
- spawnAsteroid(): void;
- spawnAlien(): void;
- spawnOxygenTank(): void;
- spawnSpeedBoost(): void;

DEMO

Kodexempel

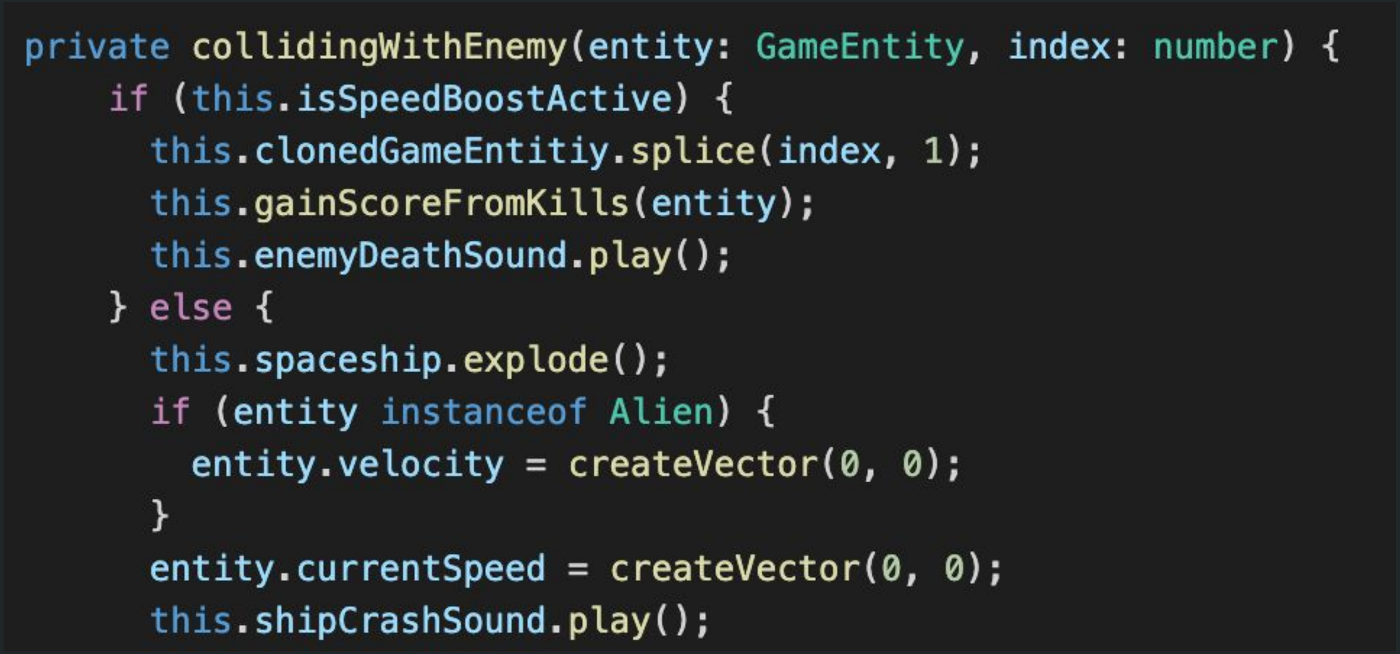


```
private checkCollision() {
    const spaceship = this.gameEntities.find((e) => instanceof SpaceShip);
    if (!spaceship) return;

    for (let i = 0; i < this.gameEntities.length; i ++) {
      const entity = this.gameEntities[i];
      if (entity === spaceship) continue;

      if (
        spaceship.position.x < entity.position.x + entity.size.x &&
        spaceship.position.x + this.spaceship.size.x > entity.position.x &&
        spaceship.position.y < entity.position.y + entity.size.y &&
        spaceship.size.y + this.spaceship.position.y > entity.position.y
      ) {
        spaceship.handleCollision(this.game);
```



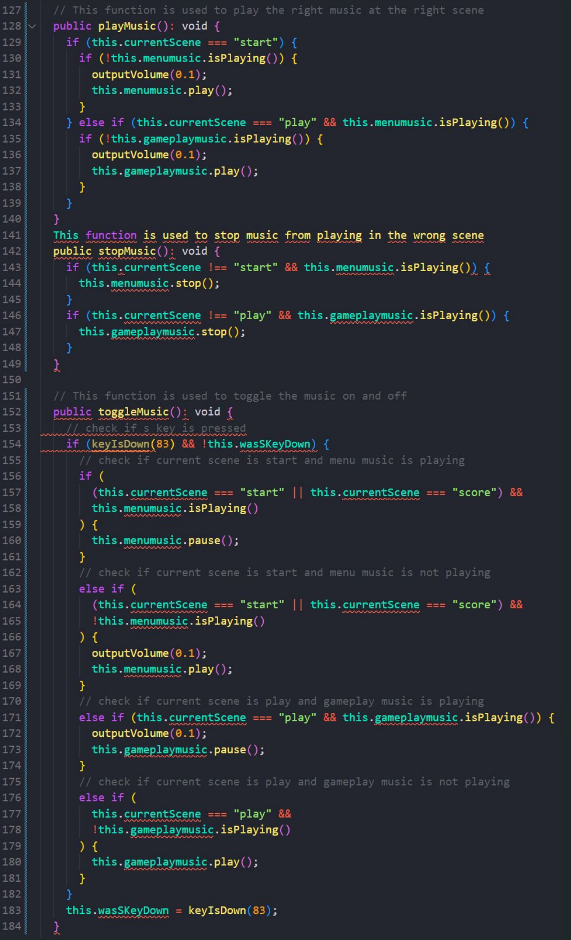
```
private checkCollision(entity: GameEntity, index: number) {
    if (
      this.spaceship.getHitBox().x < entity.getHitBox().x + entity.getHitBox().width &&
      this.spaceship.getHitBox().x + this.spaceship.getHitBox().width > entity.getHitBox().x &&
      this.spaceship.getHitBox().y < entity.getHitBox().y + entity.getHitBox().height &&
      this.spaceship.getHitBox().height + this.spaceship.getHitBox().y > entity.getHitBox().y
    ) {
      if (!(entity instanceof OxygenTank) && !(entity instanceof SpeedBoost)) {
        this.collidingWithEnemy(entity, index);
      }

      if (entity instanceof OxygenTank) {
        this.collidingWithOxygenTank(index);
      }

      if (entity instanceof SpeedBoost) {
        this.collidingWithSpeedBoost(index);
```

Kodexempel



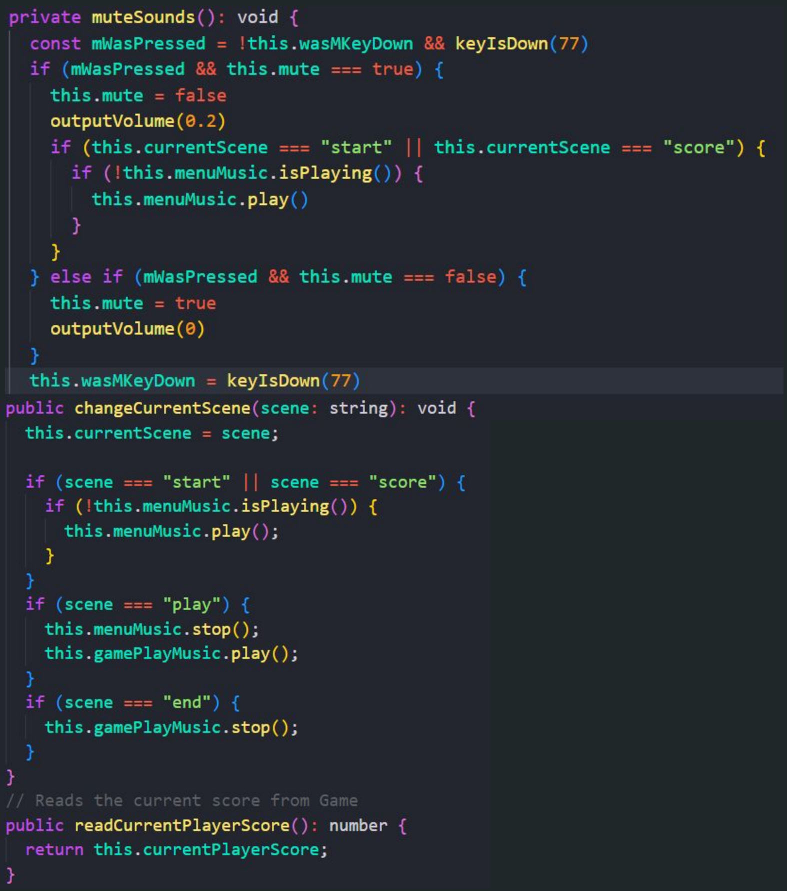
```
private collidingWithEnemy(entity: GameEntity, index: number) {
    if (this.isSpeedBoostActive) {
      this.clonedGameEntitiy.splice(index, 1);
      this.gainScoreFromKills(entity);
      this.enemyDeathSound.play();
    } else {
      this.spaceship.explode();
      if (entity instanceof Alien) {
        entity.velocity = createVector(0, 0);
      }
      entity.currentSpeed = createVector(0, 0);
      this.shipCrashSound.play();
```

muteSounds()



```
// This function is used to play the right music at the right scene
public playMusic(): void {
  if (this.currentScene === "start") {
    if (!this.menumusic.isPlaying()) {
      outputVolume(0.1);
      this.menumusic.play();
    }
  } else if (this.currentScene === "play" && this.menumusic.isPlaying()) {
    if (!this.gameplaymusic.isPlaying()) {
      outputVolume(0.1);
      this.gameplaymusic.play();
    }
  }
}
This function is used to stop music from playing in the wrong scene
public stopMusic(): void {
  if (this.currentScene !== "start" && this.menumusic.isPlaying()) {
    this.menumusic.stop();
  }
  if (this.currentScene !== "play" && this.gameplaymusic.isPlaying()) {
    this.gameplaymusic.stop();
  }
}

// This function is used to toggle the music on and off
public toggleMusic(): void {
  // check if s key is pressed
  if (keyIsDown(83) && !this.wasSKeyDown) {
    // check if current scene is start and menu music is playing
    if (
      (this.currentScene === "start" || this.currentScene === "score") &&
      this.menumusic.isPlaying()
    ) {
      this.menumusic.pause();
    }
    // check if current scene is start and menu music is not playing
    else if (
      (this.currentScene === "start" || this.currentScene === "score") &&
      !this.menumusic.isPlaying()
    ) {
      outputVolume(0.1);
      this.menumusic.play();
    }
    // check if current scene is play and gameplay music is playing
    else if (this.currentScene === "play" && this.gameplaymusic.isPlaying()) {
      outputVolume(0.1);
      this.gameplaymusic.pause();
    }
    // check if current scene is play and gameplay music is not playing
    else if (
      this.currentScene === "play" &&
      !this.gameplaymusic.isPlaying()
    ) {
      this.gameplaymusic.play();
    }
  }
  this.wasSKeyDown = keyIsDown(83);
}
```

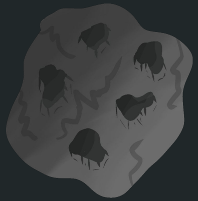
58 lines vs 33



```
private muteSounds(): void {
  const mWasPressed = !this.wasMKeyDown && keyIsDown(77)
  if (mWasPressed && this.mute === true) {
    this.mute = false
    outputVolume(0.2)
    if (this.currentScene === "start" || this.currentScene === "score") {
      if (!this.menuMusic.isPlaying()) {
        this.menuMusic.play()
      }
    }
  } else if (mWasPressed && this.mute === false) {
    this.mute = true
    outputVolume(0)
  }
  this.wasMKeyDown = keyIsDown(77)
public changeCurrentScene(scene: string): void {
  this.currentScene = scene;

  if (scene === "start" || scene === "score") {
    if (!this.menuMusic.isPlaying()) {
      this.menuMusic.play();
    }
  }
  if (scene === "play") {
    this.menuMusic.stop();
    this.gamePlayMusic.play();
  }
  if (scene === "end") {
    this.gamePlayMusic.stop();
  }
}
// Reads the current score from Game
public readCurrentPlayerScore(): number {
  return this.currentPlayerScore;
}
```

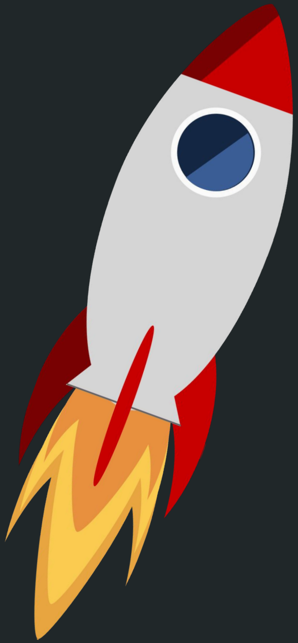
Saker som vi haft problem med

- GIT
- Live server/Vite vid live share
- GANTT-schema
- Att fastna i koden
- Svårt att hålla klass-schemat uppdaterat
- Prettier-config i projektet, hade underlättat
- Vi krånglar till det! (Mycket onödig kod men inte kod i onödan)



Saker som varit bra

- Prioritetsordning
- Parprogrammering
- Strukturering
- Samarbetet
- Dagliga morgonmöten



Tack!