Spaceplayer

Alex Rothauge
Redwan Obid
Jann Karsten
John Fleischhacker
Dominik Robin Esser

Betreuer: Daniel Wohlfarth

Gliederung

- Software Stack
- Organisation
  - Pipeline
  - Deployment
- Testing
- Frontend - React (Workflow)
- Game Backend
- Frontend - Phaser
- Code Beispiele
- Sound und Animation
- Teamarbeit
- Live Demo

Softwarestack

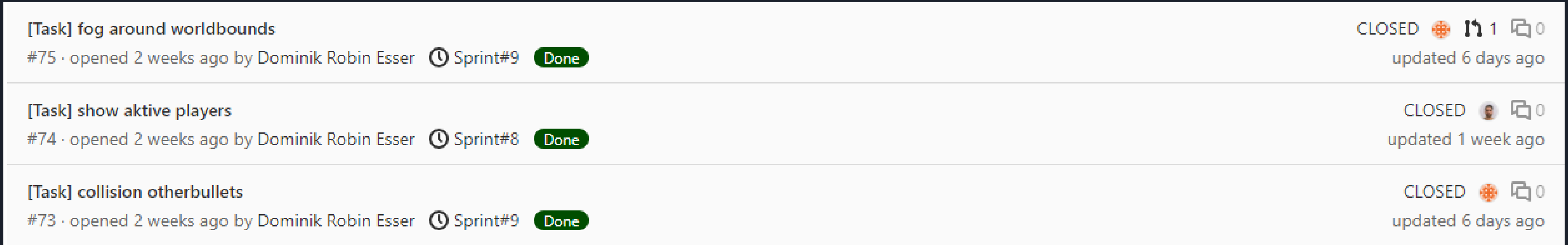
- MySQL
- Node
- Typescript
- TypeORM
- Express
- Colyseus
- React
- Phaser.io

Organisation

- Kommunikationsmittel
  - Discord
  - Whatsapp
  - Slack
- Wöchentliche Treffen
  - Sonntags im Team (Sprint Review)
  - alle 2 Wochen mittwochs mit dem Dozent
  - zusätzliche Treffen wenn nötig
- Arbeitsweise
  - Wochen Sprints
  - Issues in Gitlab
  - branches und merge Requests

Organisation

Issues

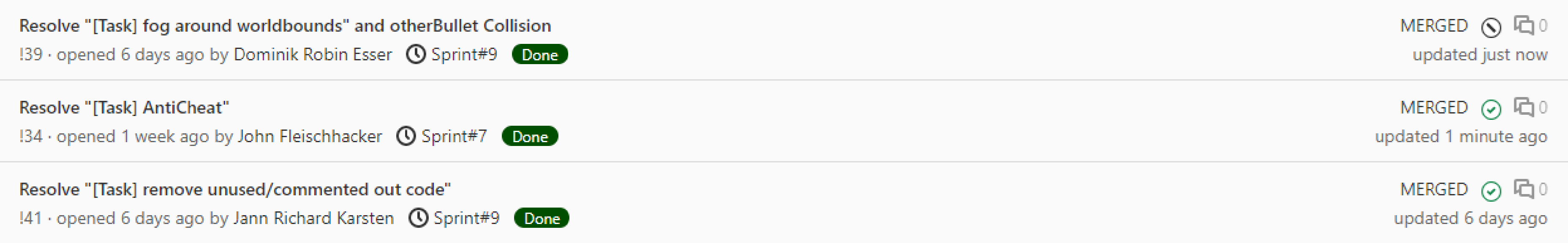


[Task] fog around worldbounds
#75 · opened 2 weeks ago by Dominik Robin Esser Sprint#9 Done
CLOSED 1 0
updated 6 days ago

[Task] show aktive players
#74 · opened 2 weeks ago by Dominik Robin Esser Sprint#8 Done
CLOSED 0
updated 1 week ago

[Task] collision otherbullets
#73 · opened 2 weeks ago by Dominik Robin Esser Sprint#9 Done
CLOSED 0
updated 6 days ago

Merge Requests

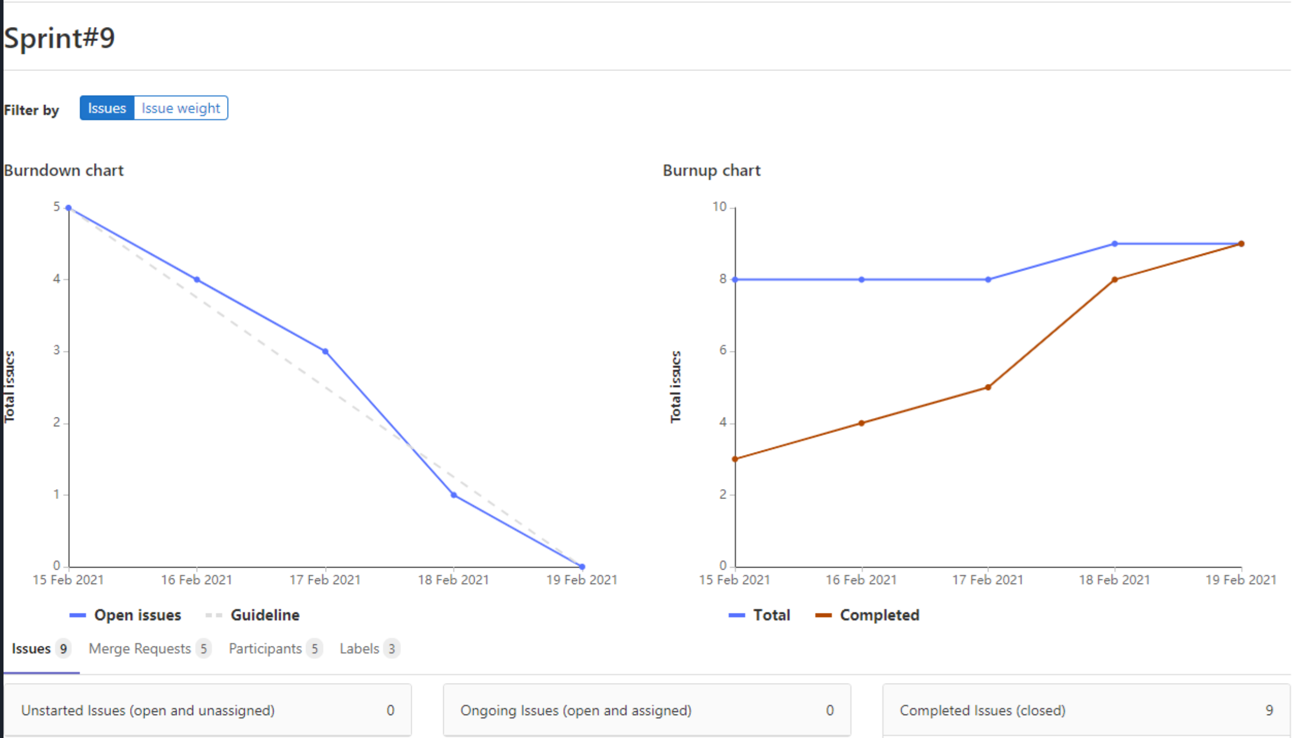


Resolve "[Task] fog around worldbounds" and otherBullet Collision
!39 · opened 6 days ago by Dominik Robin Esser Sprint#9 Done
MERGED 0
updated just now

Resolve "[Task] AntiCheat"
!34 · opened 1 week ago by John Fleischhacker Sprint#7 Done
MERGED 0
updated 1 minute ago

Resolve "[Task] remove unused/commented out code"
!41 · opened 6 days ago by Jann Richard Karsten Sprint#9 Done
MERGED 0
updated 6 days ago

Organisation

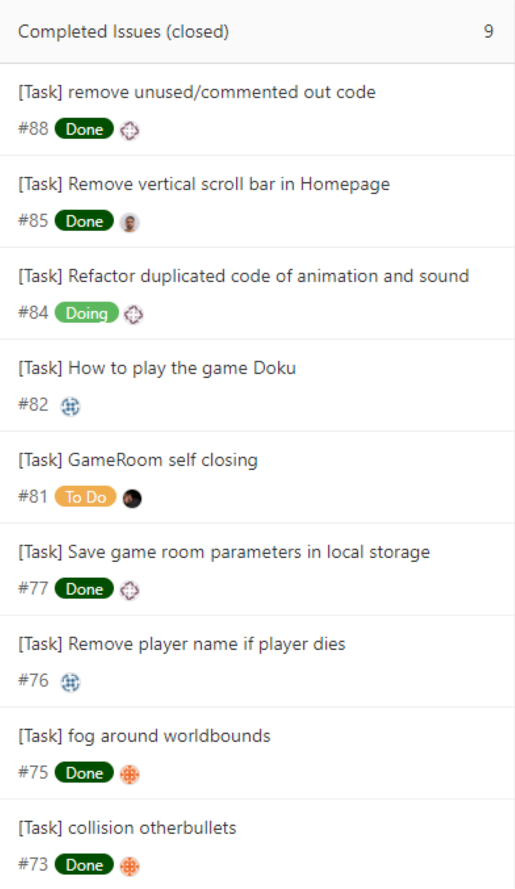


Sprint#9

Filter by **Issues** Issue weight

Issues 9 Merge Requests 5 Participants 5 Labels 3

Unstarted Issues (open and unassigned)	0	Ongoing Issues (open and assigned)	0	Completed Issues (closed)	9



Completed Issues (closed) 9

[Task] remove unused/commented out code
#88 Done

[Task] Remove vertical scroll bar in Homepage
#85 Done

[Task] Refactor duplicated code of animation and sound
#84 Doing

[Task] How to play the game Doku
#82

[Task] GameRoom self closing
#81 To Do

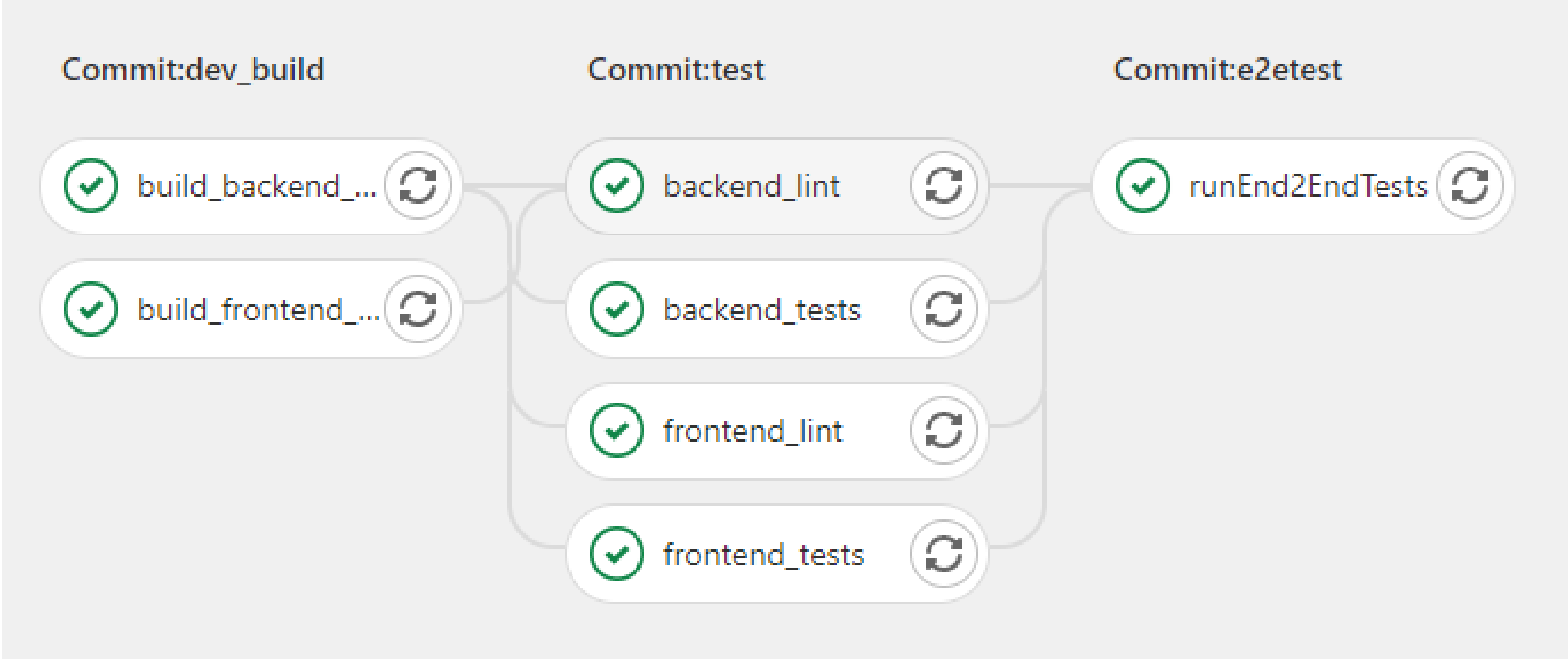
[Task] Save game room parameters in local storage
#77 Done

[Task] Remove player name if player dies
#76

[Task] fog around worldbounds
#75 Done

[Task] collision otherbullets
#73 Done

Pipeline



Commit:dev_build
Commit:test
Commit:e2etest
build_backend_...
build_frontend_...
backend_lint
backend_tests
frontend_lint
frontend_tests
runEnd2EndTests

Deployment

- Google Cloud
- einfache VM
  - e2 standard 8GB, 2CPU
  - ubuntu Minimal
  - 20GB SSD

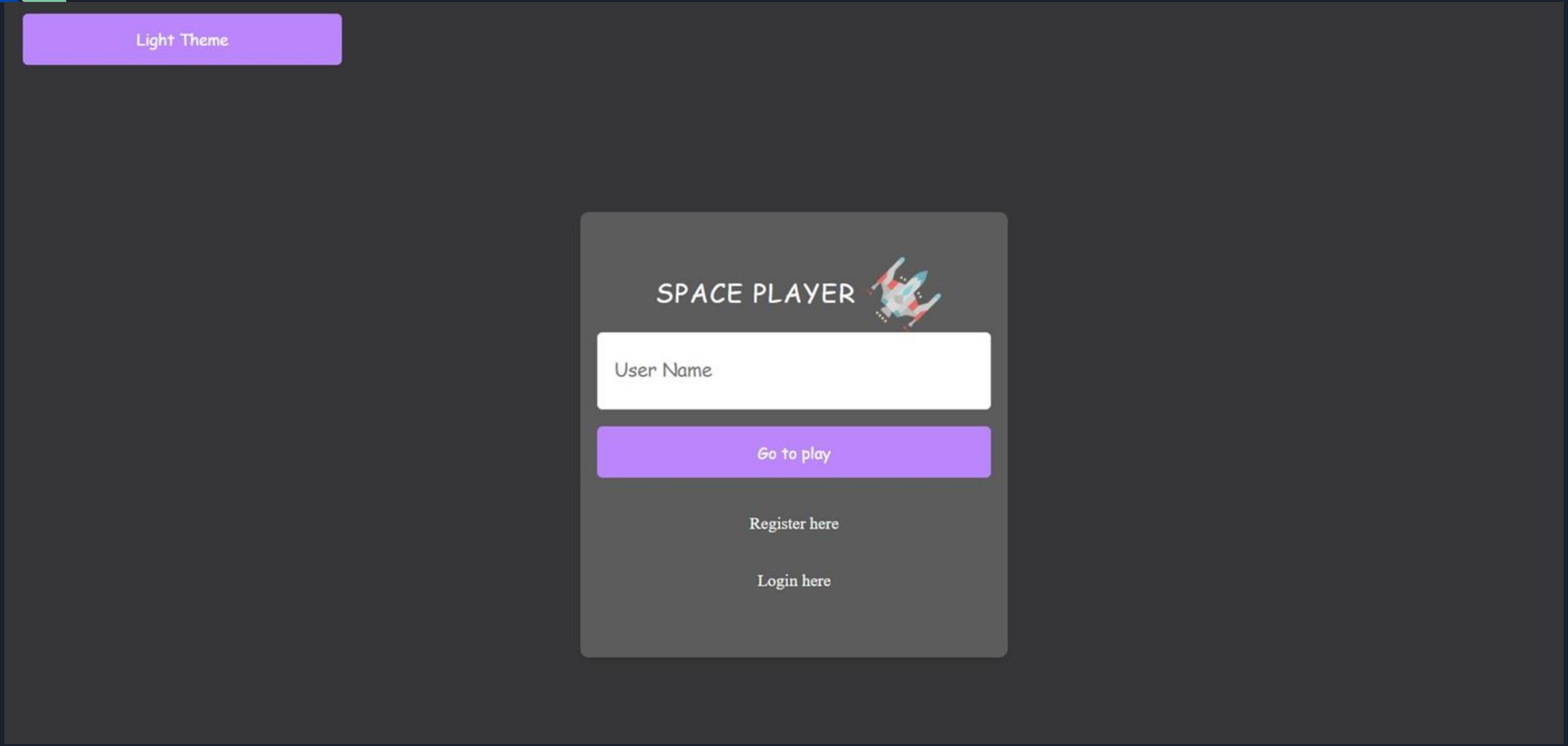
- manuelles Deployment

- Automatische Optionen
  - Kubernetes
  - Google App Engine für Nodejs

Testing

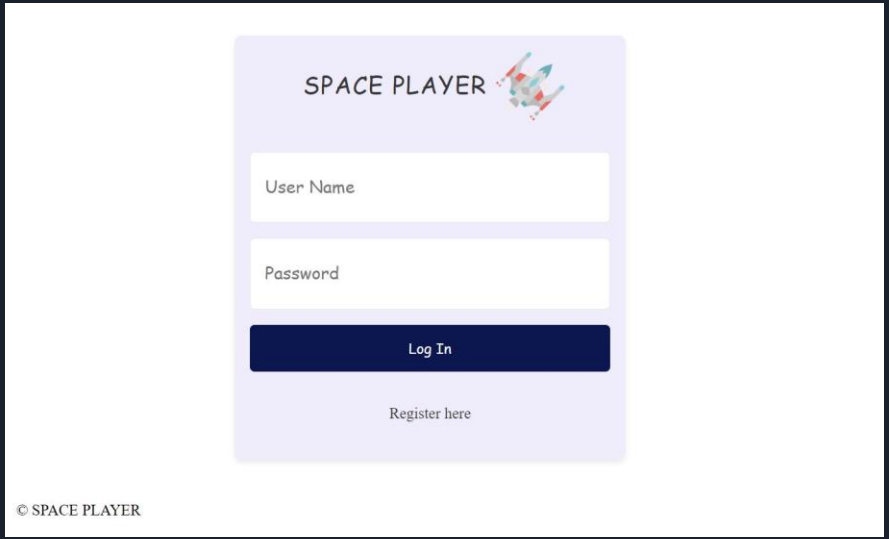
- Backend
    - Integration test mit jest
- Frontend
    - Integration test mit jest
- End 2 End Test
    - Cypress

Guest Page


Light Theme
SPACE PLAYER
User Name
Go to play
Register here
Login here

Login Page

Register Page



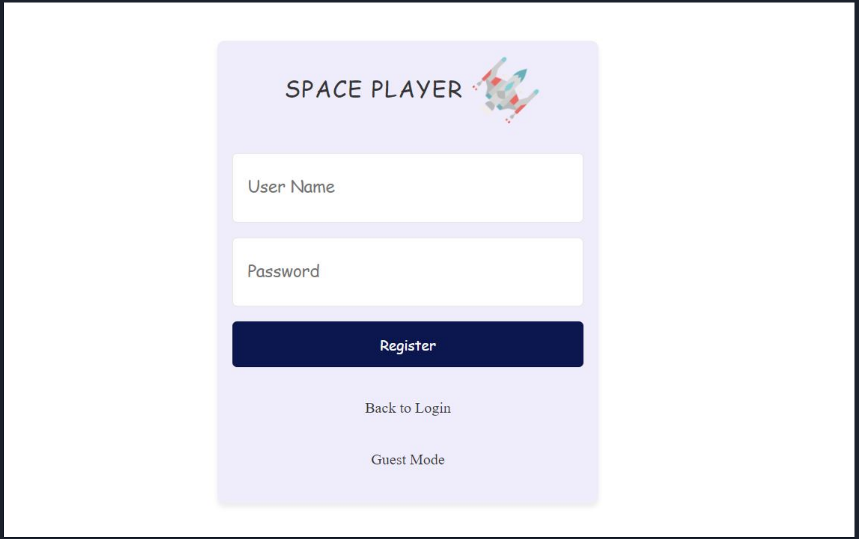
SPACE PLAYER

User Name

Password

Register here

© SPACE PLAYER



SPACE PLAYER

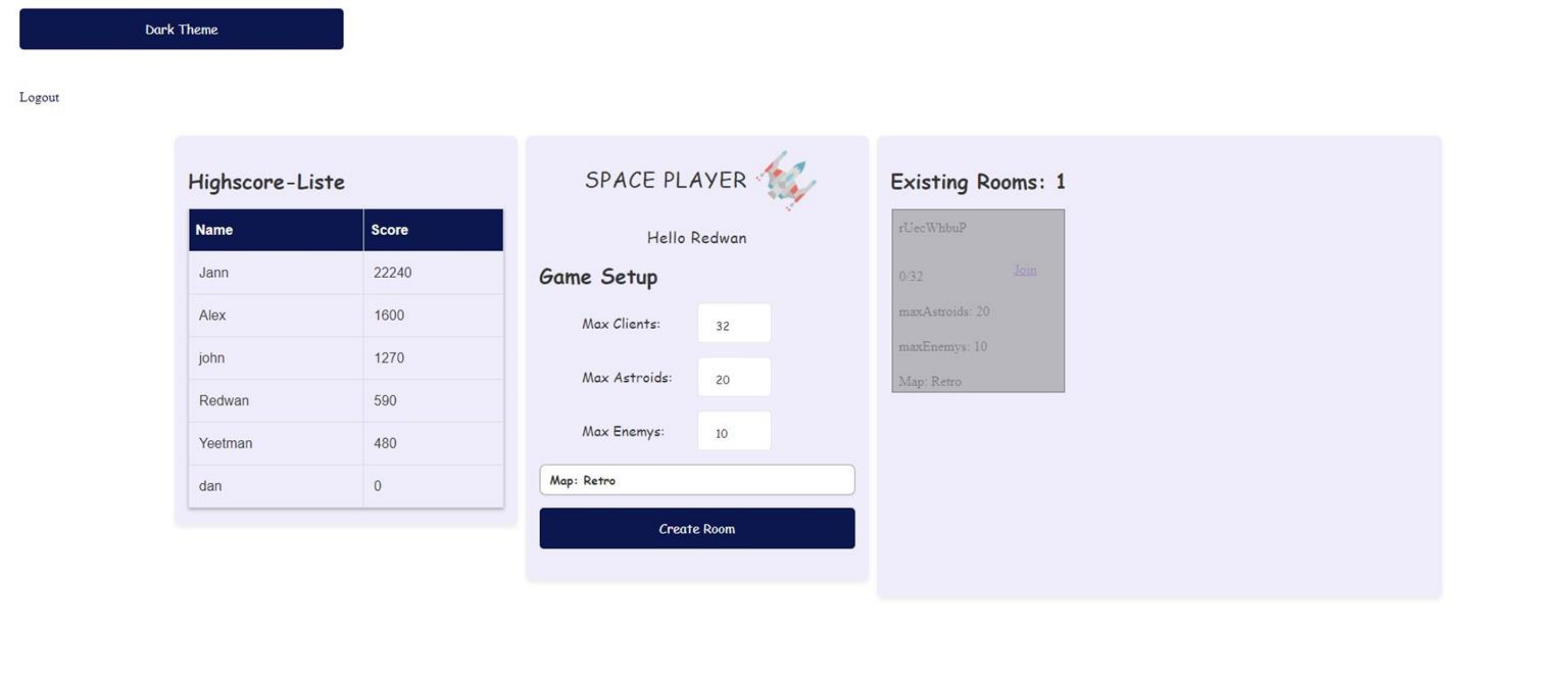
User Name

Password

Back to Login

Guest Mode

Home Page



Highscore-Liste

Name	Score
Jann	22240
Alex	1600
john	1270
Redwan	590
Yeetman	480
dan	0

SPACE PLAYER

Hello Redwan

Game Setup

Max Clients: 32

Max Astroids: 20

Max Enemys: 10

Map: Retro

Existing Rooms: 1

rUecWhbuP

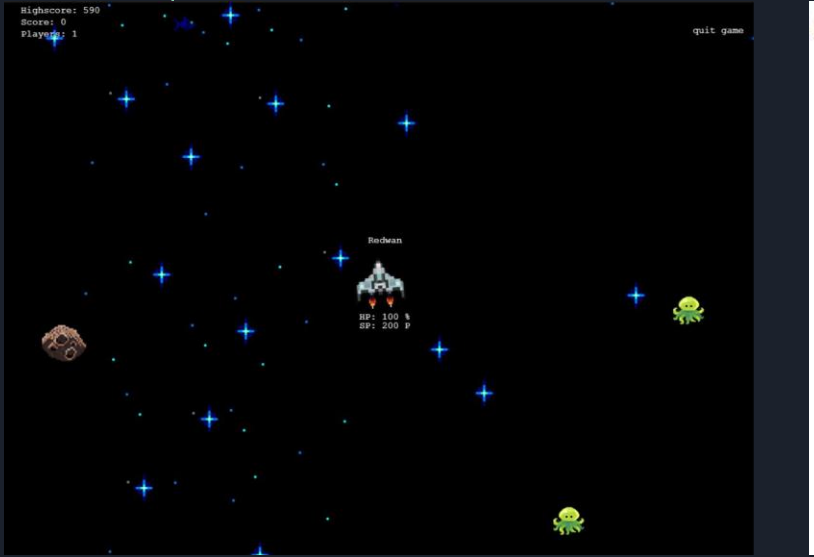
0/32

maxAstroids: 20

maxEnemys: 10

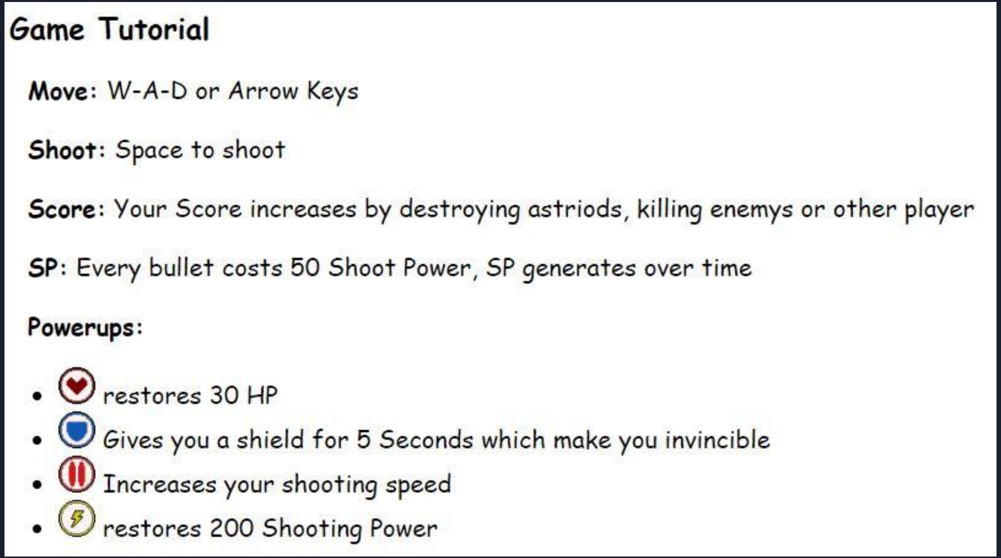
Map: Retro

Game Page



Highscore: 590
Score: 0
Players: 1

HP: 100 %
SP: 200 P



Game Tutorial

Move: W-A-D or Arrow Keys

Shoot: Space to shoot

Score: Your Score increases by destroying astriods, killing enemys or other player

SP: Every bullet costs 50 Shoot Power, SP generates over time

Powerups:

- ♥ restores 30 HP
- 🛡 Gives you a shield for 5 Seconds which make you invincible
- ⏸ Increases your shooting speed
- ⚡ restores 200 Shooting Power

Game Backend: Colyseus

- Synchronisation der Spieler und Spielwelt
- Konfigurierbares Messagingsystem
- gameServer.define('game_room', GameRoom)
- new Colyseus.Client(ws://ip-address:port)
- client.joinOrCreate('my_room')
- gamescene.client.joinById(`${gameRoomid}`)
- onCreate(_options: optionProps) {...}
- onMessage('message-key’, (client, message) => {...})
- broadcast('message-key’, data)

Game Backend: Colyseus

- `this.setState (new State ())`

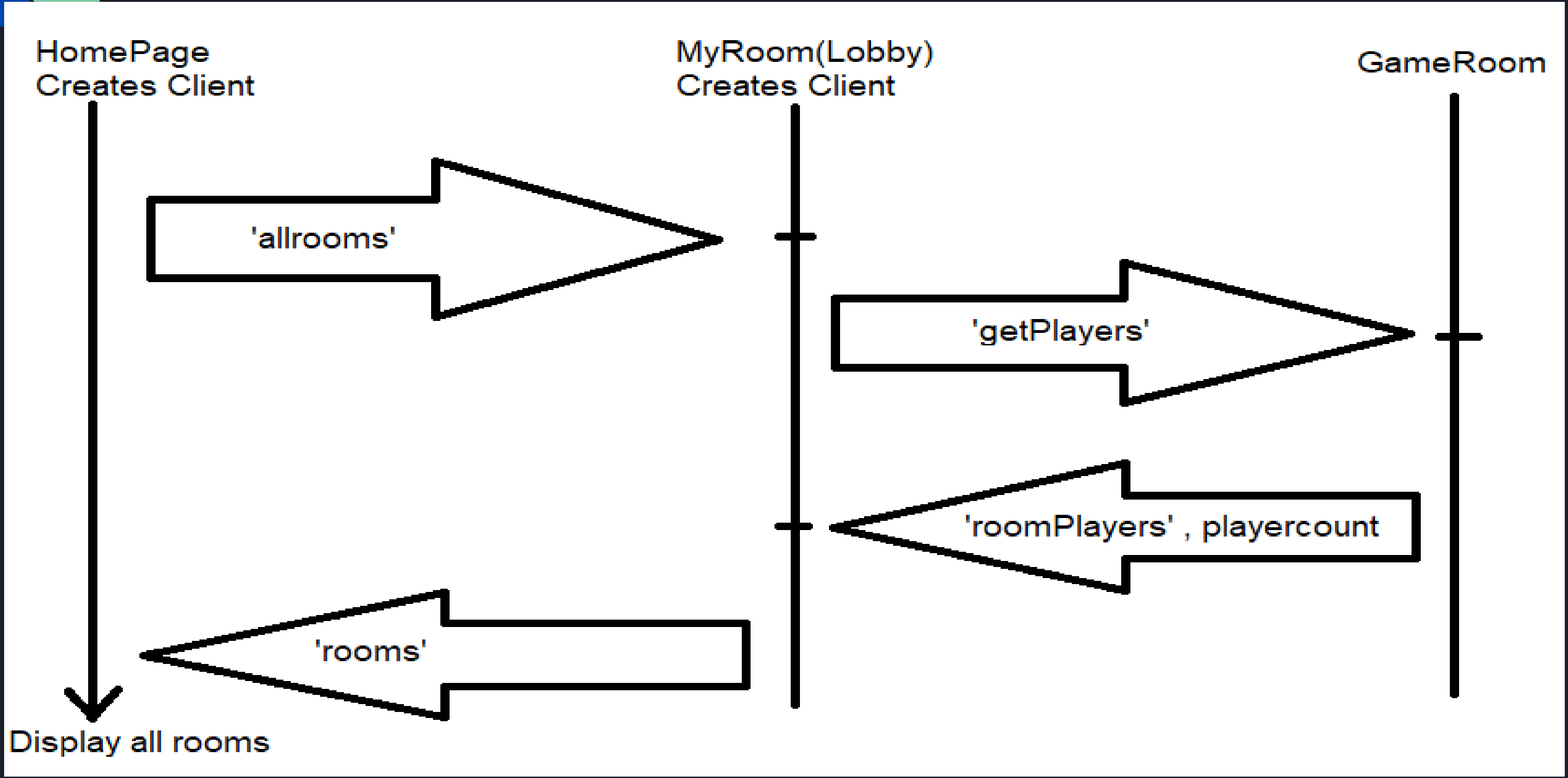
-
  ```
  class State extends Schema {
  @type({ map: Player })
  players = new MapSchema<Player>();
  ……..
  }
  ```

Game Backend: Colyseus

LobbyRoom:

- 'created'
- 'disposedRoom'
- 'allrooms'
- 'getPlayers'
- State: alle Spielräume

Game Backend: Colyseus



Game Backend: Colyseus

GameRoom:

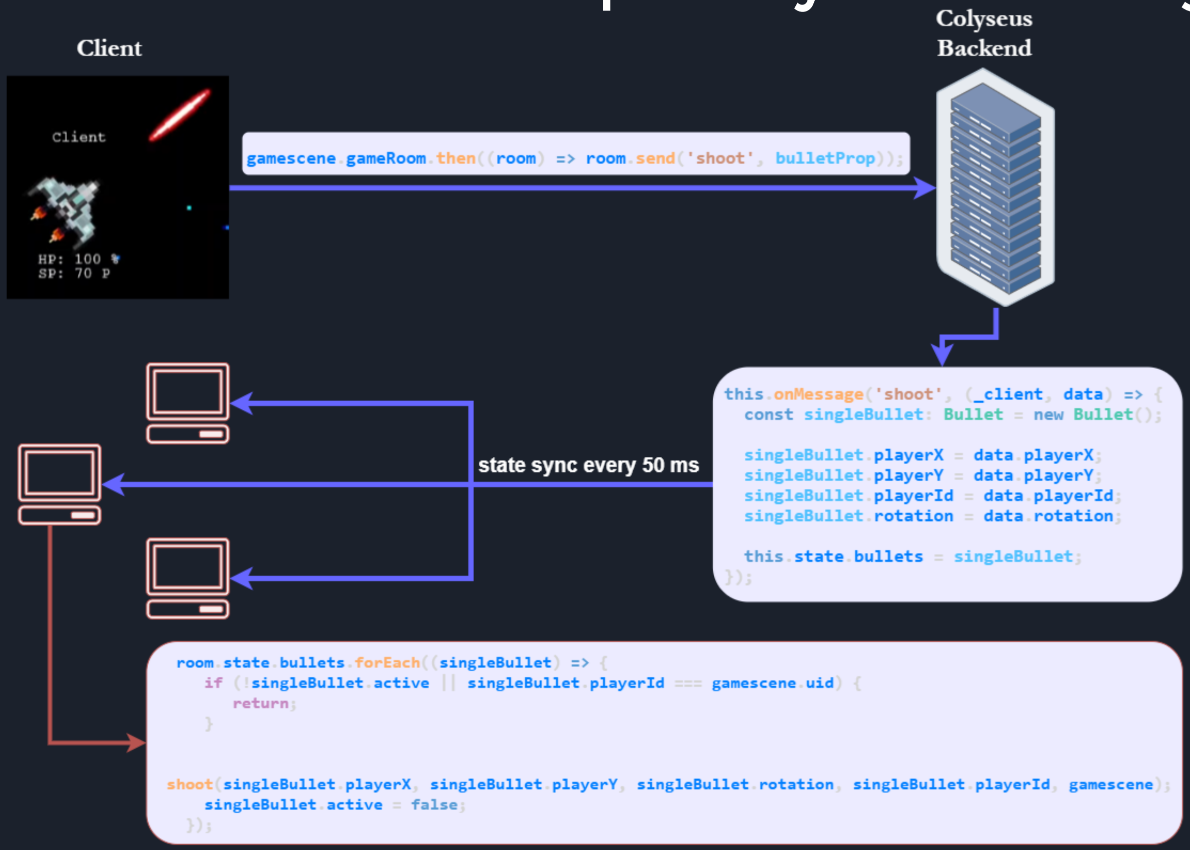
- 'roomPlayers'
- 'move'
- 'shoot'
- 'destroyAstroid'
- 'destroyShip'
- 'destroyEnemy'
- 'pickUpPowerUp'
- 'spawnPowerUp'
- 'updateScore'
- 'setHighScore'
- 'getMap'

- State: Players, Bullets, maxClients, maxAstroids, maxEnemys, astroids, enemys, powerups, map, empty, emptyTimeStamp

Game Frontend: Phaser

- hinzufügen/entfernen von Sprites mit Körpern
- gamescene.physics.add.sprite(..)
- sprite.destroy()
- KeyListener z.B. ‘if (space.isDown)’
- Welt und Grenzen
- physics.world.setBounds(...)
- player.body.setCollideWorldBounds(true)
- Hintergrundbild durch add.tileSprite(..)
- Kamera folgt dem Spieler
- cameras.main.startFollow(...)
- Collision (Groups) overlap und collide
- asteroids = add.group()
- physics.overlap(bulletsGroup,asteroids)
- Raumschiffbewegung (rotation, veloctiy, drag)

Game Backend Beispiel: Sync Shooting



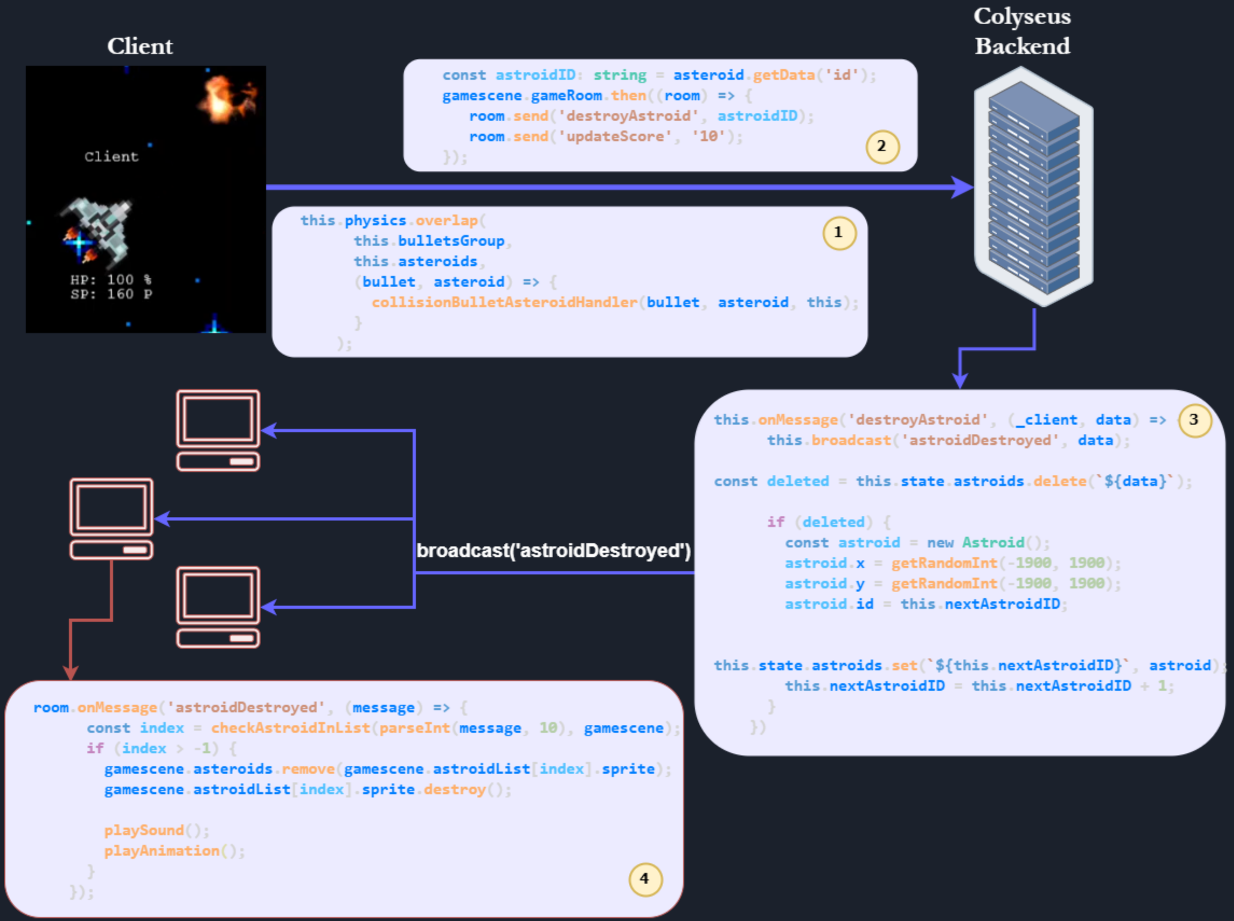
```
gamescene.gameRoom.then((room) => room.send('shoot', bulletProp));
```

```
this.onMessage('shoot', (_client, data) => {
  const singleBullet: Bullet = new Bullet();

  singleBullet.playerX = data.playerX;
  singleBullet.playerY = data.playerY;
  singleBullet.playerId = data.playerId;
  singleBullet.rotation = data.rotation;

  this.state.bullets = singleBullet;
});
```

```
 room.state.bullets.forEach((singleBullet) => {
   if (!singleBullet.active || singleBullet.playerId === gamescene.uid) {
     return;
   }


shoot(singleBullet.playerX, singleBullet.playerY, singleBullet.rotation, singleBullet.playerId, gamescene);
   singleBullet.active = false;
  });
```

update every 17 ms (60 FPS)

Game Backend Beispiel: Destroy Astroid



1

```
this.physics.overlap(
      this.bulletsGroup,
      this.asteroids,
      (bullet, asteroid) => {
        collisionBulletAsteroidHandler(bullet, asteroid, this);
      }
    );
```

2

```
const astroidID: string = asteroid.getData('id');
gamescene.gameRoom.then((room) => {
   room.send('destroyAstroid', astroidID);
   room.send('updateScore', '10');
});
```

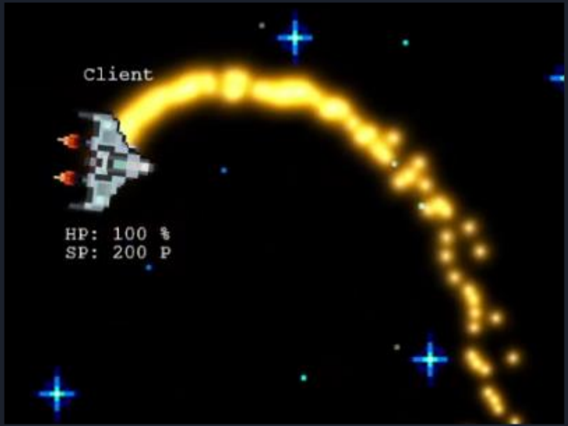
3

```
this.onMessage('destroyAstroid', (_client, data) => {
      this.broadcast('astroidDestroyed', data);

const deleted = this.state.astroids.delete(`${data}`);

      if (deleted) {
        const astroid = new Astroid();
        astroid.x = getRandomInt(-1900, 1900);
        astroid.y = getRandomInt(-1900, 1900);
        astroid.id = this.nextAstroidID;


this.state.astroids.set(`${this.nextAstroidID}`, astroid);
        this.nextAstroidID = this.nextAstroidID + 1;
      }
    })
```

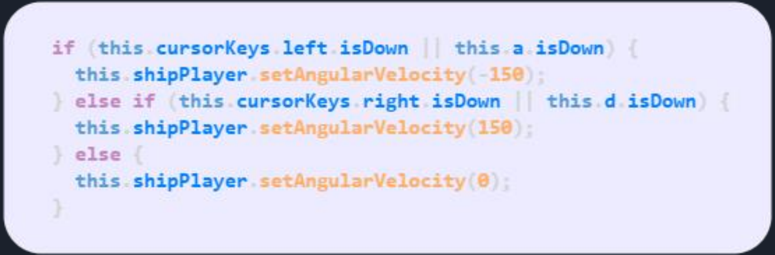
4

```
room.onMessage('astroidDestroyed', (message) => {
      const index = checkAstroidInList(parseInt(message, 10), gamescene);
      if (index > -1) {
        gamescene.asteroids.remove(gamescene.astroidList[index].sprite);
        gamescene.astroidList[index].sprite.destroy();

        playSound();
        playAnimation();
      }
    });
```

Phaser Beispiel: Ship movement physics

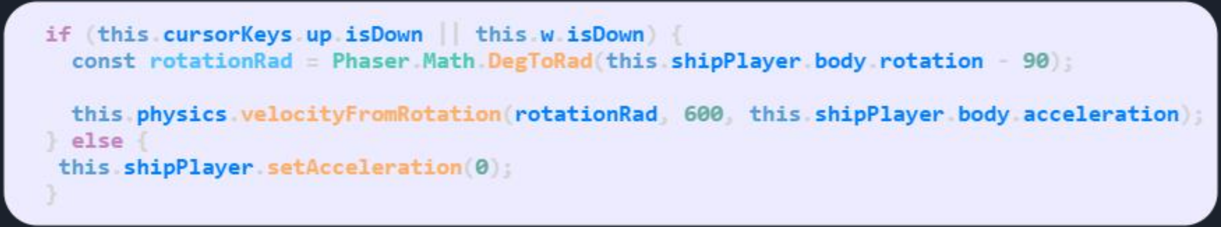


Rotation



```
if (this.cursorKeys.left.isDown || this.a.isDown) {
  this.shipPlayer.setAngularVelocity(-150);
} else if (this.cursorKeys.right.isDown || this.d.isDown) {
  this.shipPlayer.setAngularVelocity(150);
} else {
  this.shipPlayer.setAngularVelocity(0);
}
```

Movement

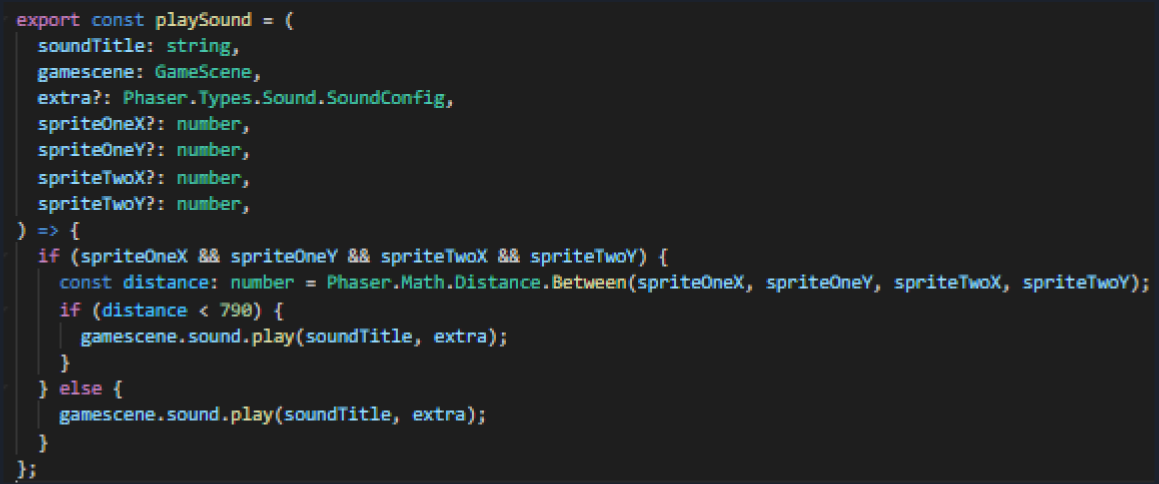


```
if (this.cursorKeys.up.isDown || this.w.isDown) {
  const rotationRad = Phaser.Math.DegToRad(this.shipPlayer.body.rotation - 90);

  this.physics.velocityFromRotation(rotationRad, 600, this.shipPlayer.body.acceleration);
} else {
 this.shipPlayer.setAcceleration(0);
}
```

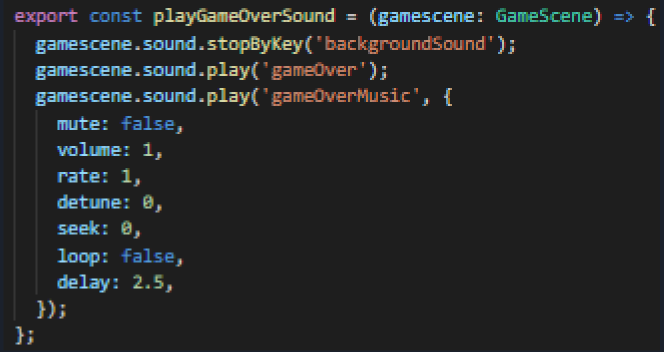
Sound und Animation: Game Sounds

Sounds für: Schüsse, Hintergrund, Game over, Zerstörungen(Qualle, Asteroid, Schiffsexplosion)

```
room.onMessage('astroidDestroyed', (message) => {...}); (‘jellyfishDied’)
```



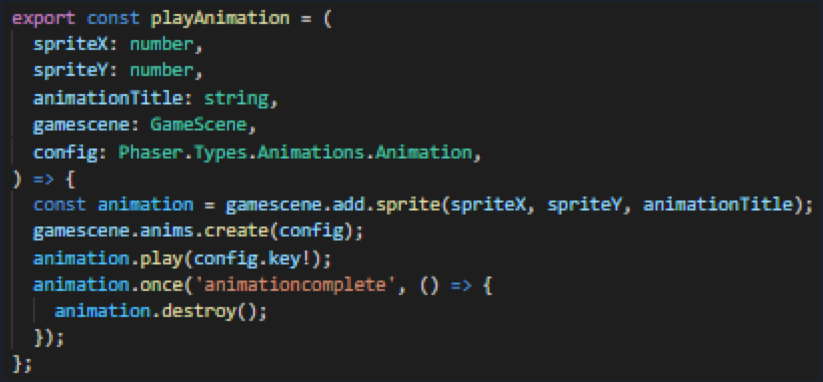
```
export const playSound = (
  soundTitle: string,
  gamescene: GameScene,
  extra?: Phaser.Types.Sound.SoundConfig,
  spriteOneX?: number,
  spriteOneY?: number,
  spriteTwoX?: number,
  spriteTwoY?: number,
) => {
  if (spriteOneX && spriteOneY && spriteTwoX && spriteTwoY) {
    const distance: number = Phaser.Math.Distance.Between(spriteOneX, spriteOneY, spriteTwoX, spriteTwoY);
    if (distance < 790) {
      gamescene.sound.play(soundTitle, extra);
    }
  } else {
    gamescene.sound.play(soundTitle, extra);
  }
};
```



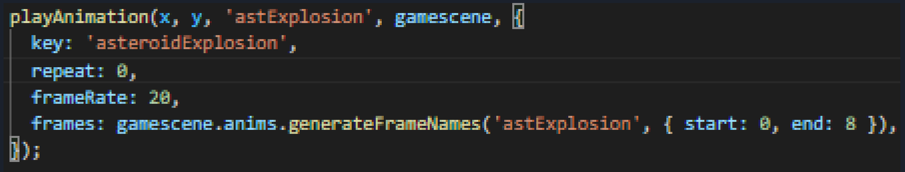
```
export const playGameOverSound = (gamescene: GameScene) => {
  gamescene.sound.stopByKey('backgroundSound');
  gamescene.sound.play('gameOver');
  gamescene.sound.play('gameOverMusic', {
    mute: false,
    volume: 1,
    rate: 1,
    detune: 0,
    seek: 0,
    loop: false,
    delay: 2.5,
  });
};
```

Sound und Animation: Game Animation mit Spritesheets

Animationen für: Zerstörungen(Qualle, Asteroid, Schiff), Nebel



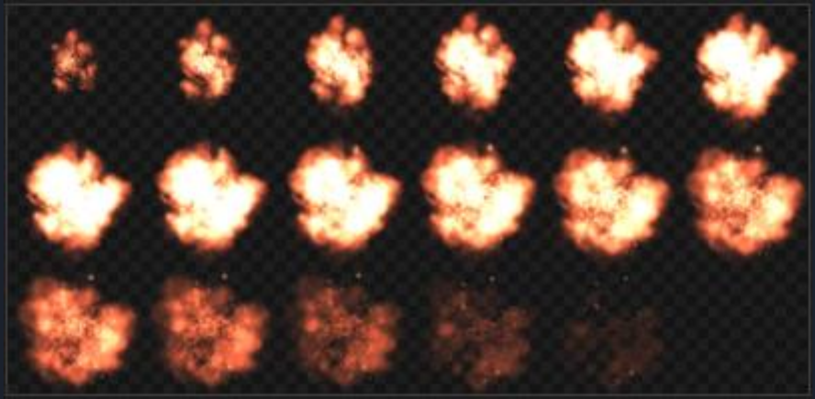
```
export const playAnimation = (
  spriteX: number,
  spriteY: number,
  animationTitle: string,
  gamescene: GameScene,
  config: Phaser.Types.Animations.Animation,
) => {
  const animation = gamescene.add.sprite(spriteX, spriteY, animationTitle);
  gamescene.anims.create(config);
  animation.play(config.key!);
  animation.once('animationcomplete', () => {
    animation.destroy();
  });
};
```



```
playAnimation(x, y, 'astExplosion', gamescene, {
  key: 'asteroidExplosion',
  repeat: 0,
  frameRate: 20,
  frames: gamescene.anims.generateFrameNames('astExplosion', { start: 0, end: 8 }),
});
```

Sound und Animation: Wie wurden die Spritesheets erstellt?

1. Spritesheets suchen
2. Image Magick Software zur Erstellung von Vektor- und Rastergrafiken
3. https://gammafp.com/tools/ wurde ein neues Spritesheet erstellt



Team Arbeit

Frontend-React: Alex, Jann, Redwan

Frontend-Phaser: Robin, Alex, Redwan, Jann, John

Backend-Game: Robin, Alex

Backend-API: John, Redwan

Testing: John, Redwan

Pipeline: John

Documentation: Alex, Redwan

Art-Work: Robin(Bullets,PowerUps,Shield)