



TONY TRASH

ECKHOFF, JOSEHPINE

ROSENKRANZ, MARC

JAHJA, ARDIAN

SCHMIDT, FELIX

DEMO-VIDEO DER ANWENDUNG



VERANTWORTUNGSBEREICHE

- Josephine:

- Grundsätzliche Konzeption und Programmierung von Hindernissen

- Marc:

- Grundlegende Spielmechaniken & Leveldesign

- Ardian:

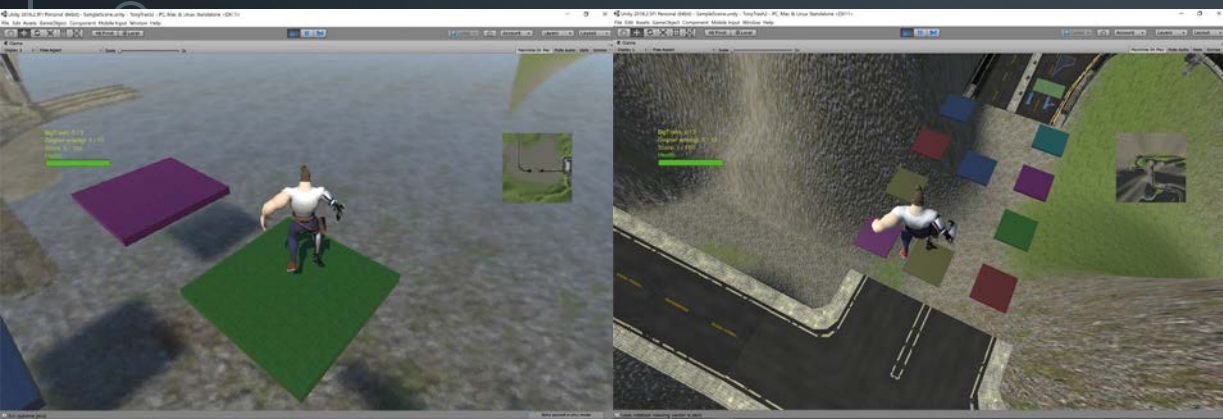
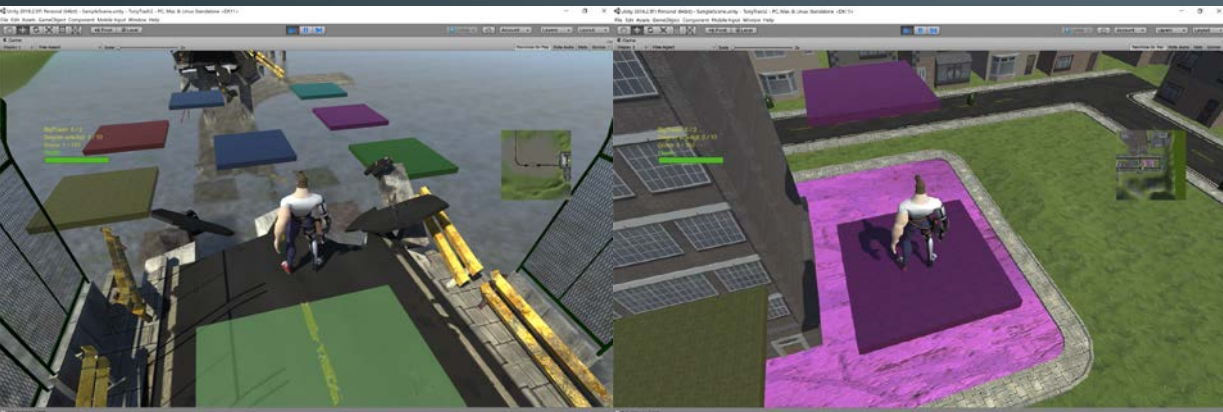
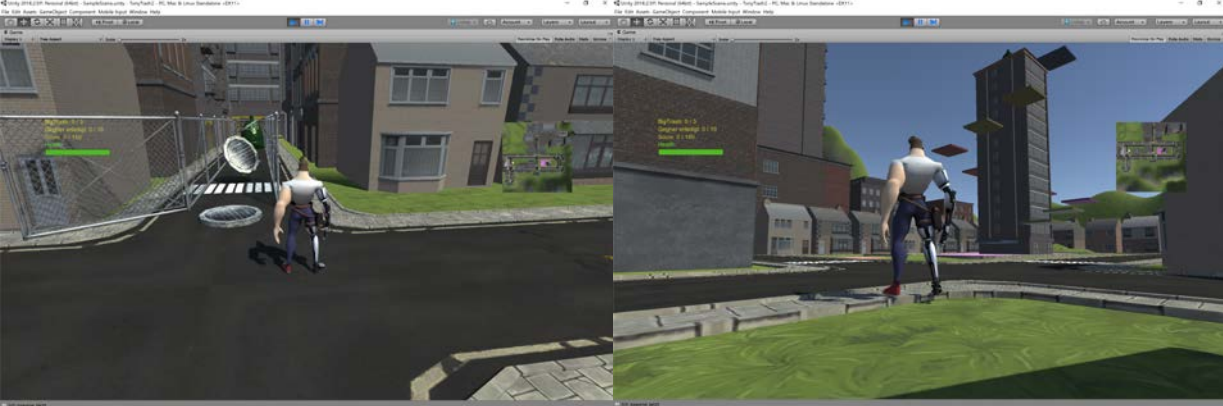
- Interaktionen und Levelmanagement

- Felix:

- Grundsätzliche Konzeption und Programmierung von Gegnern & Boss

JOSEPHINE

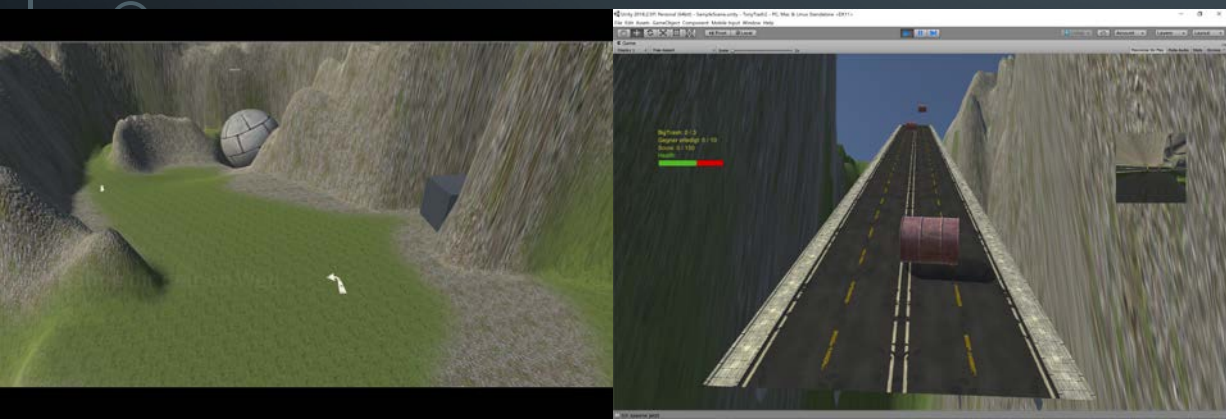
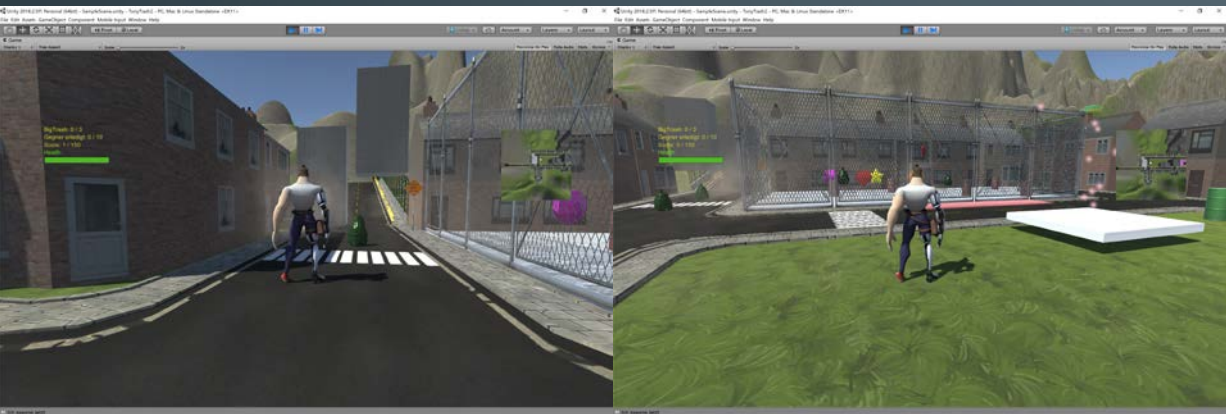
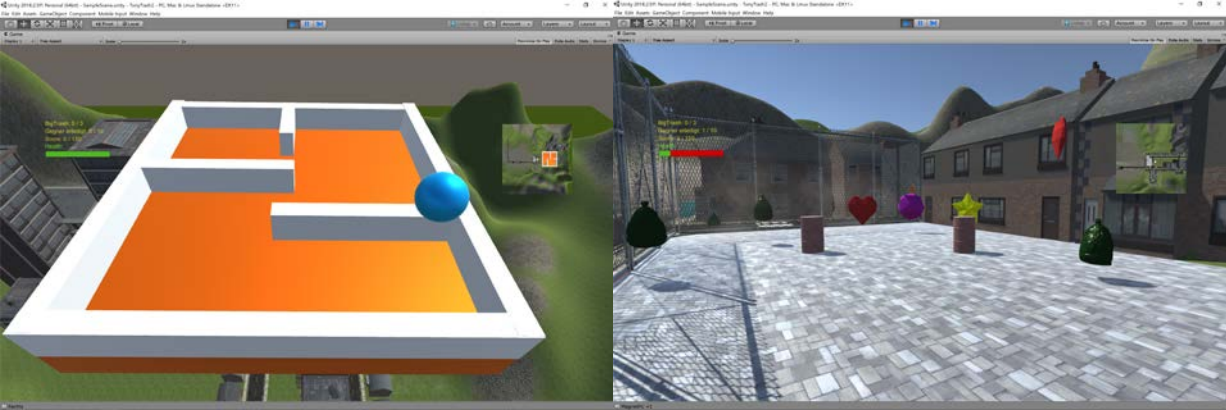
- Grundsätzliche Konzeption und Programmierung von Hindernissen
 - Plattformen in verschiedenen Ausführungen:
 - Vertikale und horizontale Bewegung
 - Zerstörung auf verschieden Arten
 - Größenveränderung
 - Verschiedene Rotationen
 - Springende Gullideckel
 - Schaden bei Berührung
 - Passierbar durch Sprünge



ARDIAN

• Interaktion und Levelmanagement

- GUI (Minimap)
- Worldeffects (Weather)
- Levelmanager
- Obstacles
 - Gates, Smasher, BarrelSpawner
- Quests
 - Minigame & Rampquest
- PowerUps
 - Health, Bomb, Jump, Magnet
- Inventarsystem



MARC

- Grundlegende Spielmechaniken & Leveldesign

- Cameracontroller

- Playercontroller

- Tod & Leben

- Health, Respawn, Playerdamage, Checkpoints

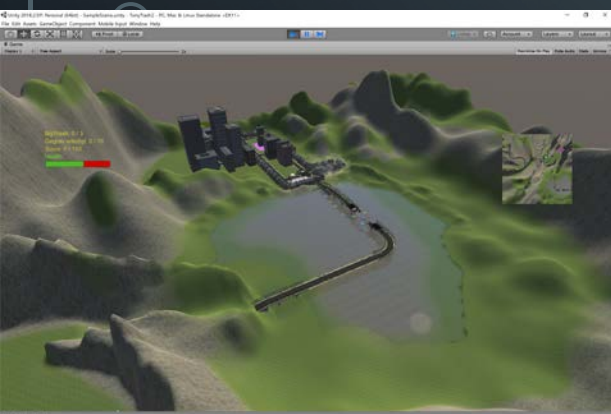
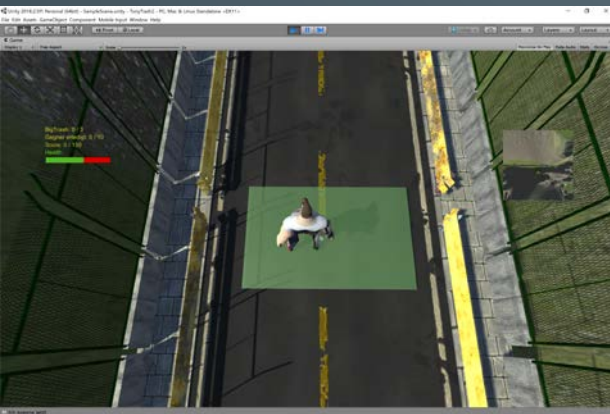
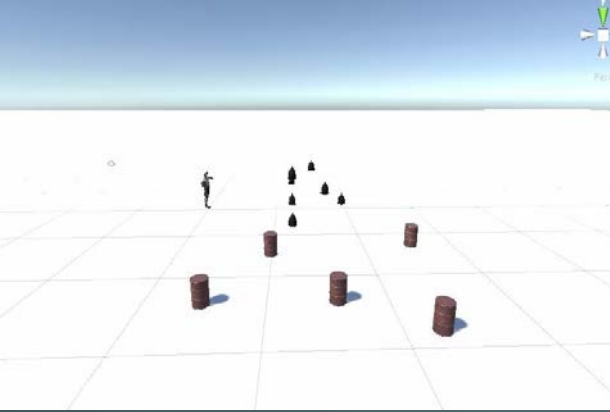
- Spawntruck & Itemspawn

- Grundlegendes Leveldesign

- Urbanes Setting

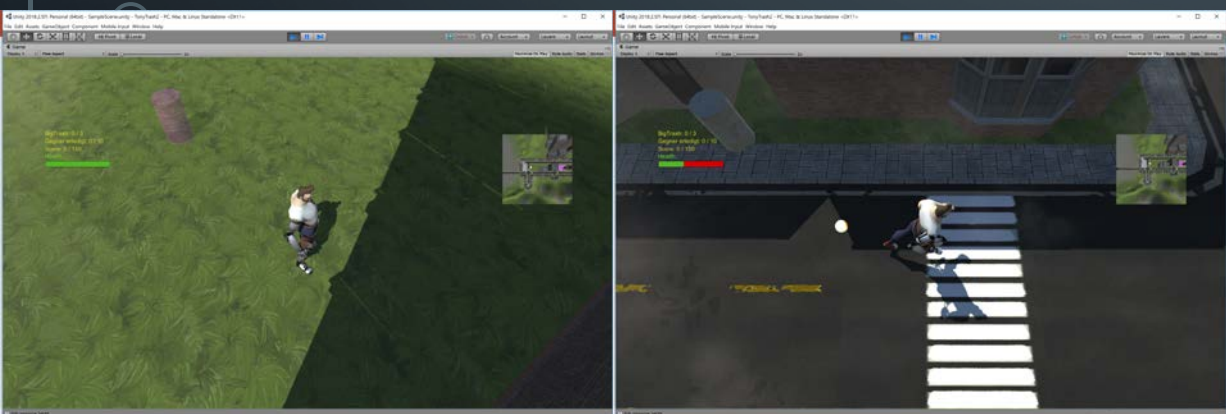
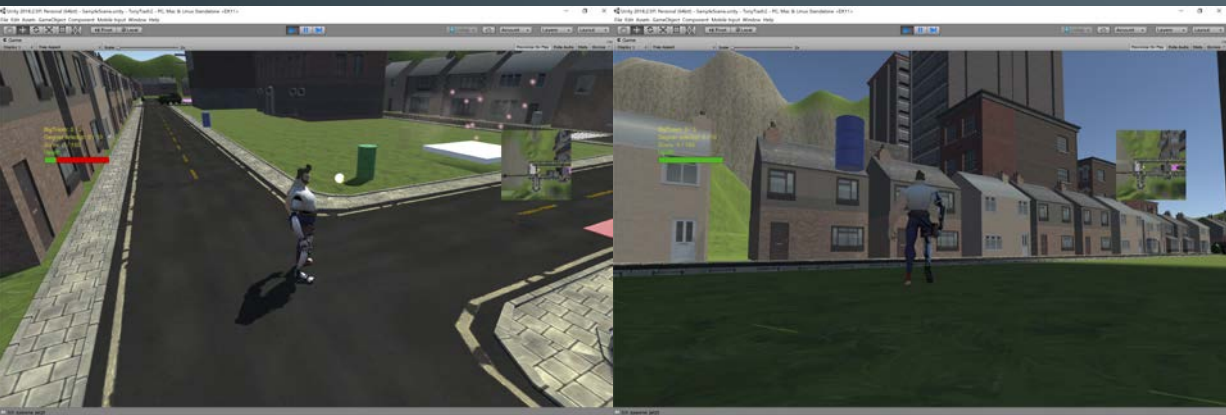
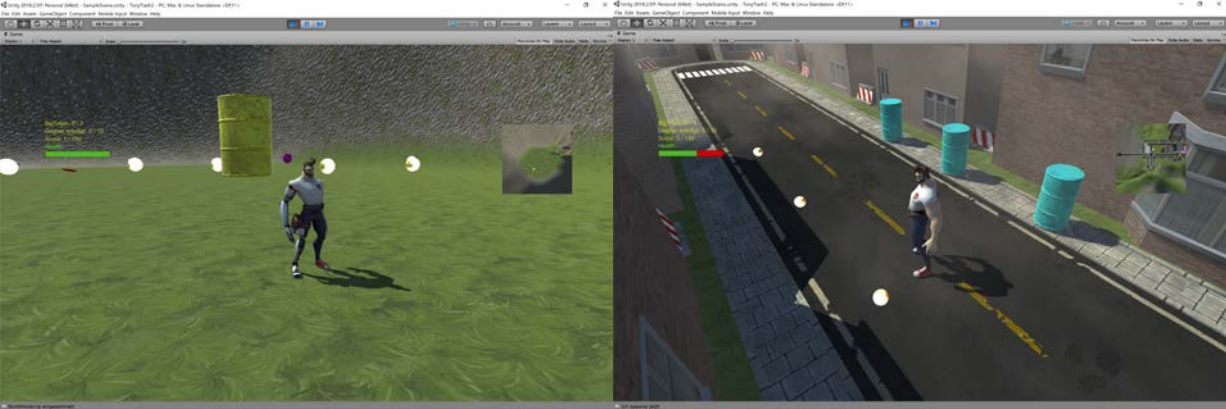
- Gebirge

- Giftteich

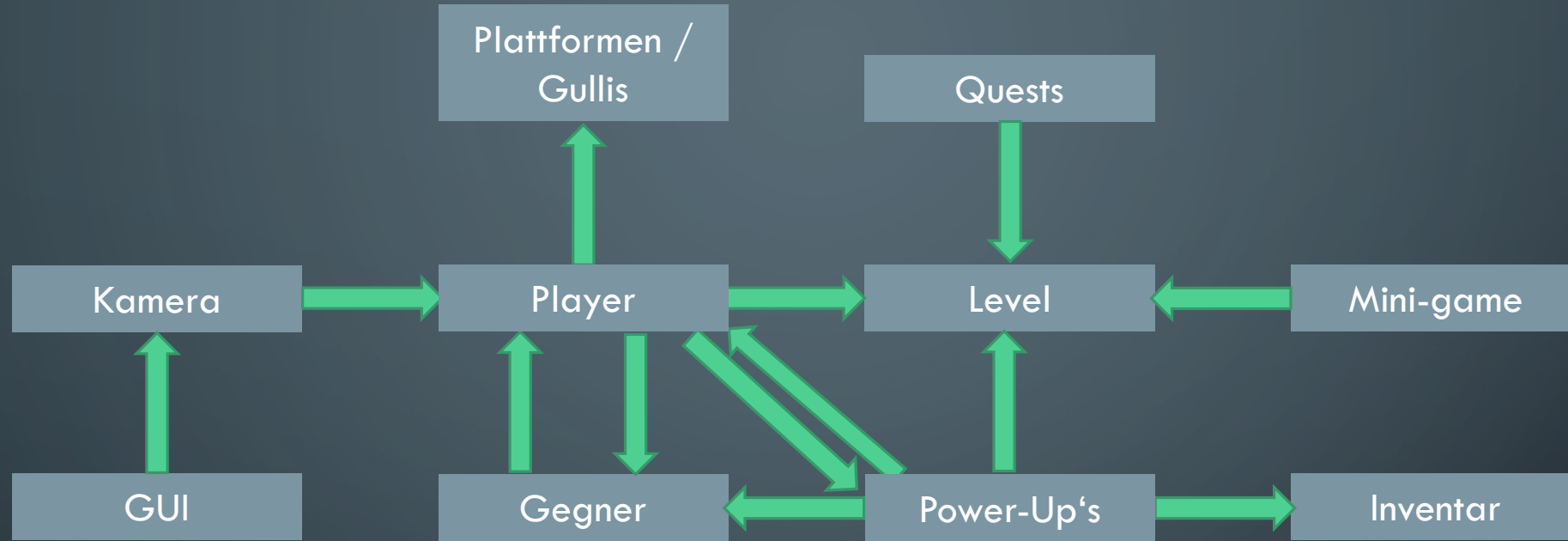


FELIX

- Grundsätzliche Konzeption und Programmierung von Gegnern und dem Boss
- Gegner in verschiedenen Ausführungen:
 - Schießende:
 - Verfolgungsschuss, Direktschuss, Zufallsschuss
 - Verfolgende:
 - Unsichtbar, Springend
- Der finale Boss:
 - Vereint die Fähigkeiten der kleineren Gegner
 - Verfolgungsschuss, Zufallsschuss, Springend



SYSTEM-ARCHITEKTUR



Klassen

Kamera: siehe Kamera Ordner

GUI: siehe Minimap Ordner, LevelOneManager, Health

Plattformen / Gullis: siehe Plattformen / Gulli Ordner

Player: siehe PlayerController Ordner

Gegner: siehe Gegner Ordner

Quests: siehe RampQuest Ordner, Gate Ordner, Smasher Ordner

Level: siehe BarrelObstacle Ordner, Itemspawn Ordner, Level Manager Ordner, Weather System Ordner

Minigame: siehe MiniGame Ordner

Power-Up's: siehe PowerUps Ordner

Inventar: siehe Inventar Ordner

ZEITPLAN SOLL

	Simon Evers	Felix Schmidt	Marc Rosenkranz	Ardian Jahja	Josephine Eckhoff
25.06. - 01.07.	Recherche GUI	Recherche KI	Recherche Assets/Texturing	Recherche Animation/ Sounds	Recherche Import von 3D-Modellen in Unity
02.07. - 08.07.	Klausurvorbereitung	Klausurvorbereitung	Klausurvorbereitung	Klausurvorbereitung	Klausurvorbereitung
09.07. - 15.07.	Klausurvorbereitung	Klausurvorbereitung	Klausurvorbereitung	Klausurvorbereitung	Klausurvorbereitung
16.07. - 22.07.	Klausurvorbereitung	Klausurvorbereitung	Klausurvorbereitung	Klausurvorbereitung	Klausurvorbereitung
23.07. - 29.07.	Klausurvorbereitung	Klausurvorbereitung	Klausurvorbereitung	Klausurvorbereitung	Klausurvorbereitung
30.07. - 05.08.	Platzhalter erstellen / grobes Leveldesign	Tutorials / grobe Verhaltensstruktur der Gegner	Kamera / Bewegung Spielfigur	Tutorials / visuelle Effekte mit Platzhaltern	Plattformen erstellen / Bewegung der Plattformen
06.08. - 12.08. MS01	Bodys und Collision für Welt erstellen	Gegnerreaktion auf Welt	3rd Person Verfolgung bei Bewegung	Anpassung visueller Effekte an Spielfigur	Kollision Platten mit Spielfigur
13.08. - 19.08.	Verknüpfung von Levelementen	Interaktion / Reaktion auf Spieler	Body und Collision für Spieler erstellen	Welteffekte	Kollisionsverhalten Platten mit Spielfigur
20.08. - 26.08. MS02	Texturierung / Feinschliff	Verfeinern von Ridgetbodies und Collisions	Aktionsspektrum der Spielfigur	Spieler-/Gegnereffekte	Kollisionsverhalten Platten mit Spielfigur
27.08. - 02.09. MS03	Bugfixing	Bugfixing	Bugfixing	Bugfixing	Bugfixing

ZEITPLAN IST

	Felix Schmidt	Marc Rosenkranz	Ardian Jahja	Josephine Eckhoff
25.06. - 01.07.	Recherche KI	Recherche Assets/Texturing	Recherche Animation/ Sounds	Recherche Import von 3D-Modellen in Unity
02.07. - 08.07.	Klausurvorbereitung	Klausurvorbereitung	Klausurvorbereitung	Klausurvorbereitung
09.07. - 15.07.	Klausurvorbereitung	Klausurvorbereitung	Klausurvorbereitung	Klausurvorbereitung
16.07. - 22.07.	Klausurvorbereitung	Klausurvorbereitung	Klausurvorbereitung	Klausurvorbereitung
23.07. - 29.07.	Klausurvorbereitung	Klausurvorbereitung	Klausurvorbereitung	Klausurvorbereitung
30.07. - 05.08.	Analyse und Ideenfindung über Verhalten von 3D Jump'n'Run Gegnern	Player- & Cameracontroller	Power-Ups & Manager	Erstellen erster Plattformen
06.08. - 12.08.	Tutorialvideos und erste Tests in eigenem Projekt	CamSwitch & FPSCam Scripte	Inventarsystem	Zerbrechende Plattform, Hide&Show Plattform
13.08. - 19.08.	Heimurlaub	Allgemeine Level & Spielmechaniken	Stampfer	Bewegende Plattformen
20.08. - 26.08.	Heimurlaub	Einfügen von Leben und Tod	Kugel Mini-Game	Verswindende Plattform
27.08. - 02.09. MS01	Einfügen von Follower, Shooter und erstem Bossprototypen	Leveldesign (Aufgrund von inaktivem 5. Gruppenmitglied)	Level-One Manager	Umzug aufgrund des anstehenden Praktikums
03.09. - 09.09.	Einfügen von Invisible, Direct- und Followshooter	Leveldesign (Aufgrund von inaktivem 5. Gruppenmitglied)	Minimap und Wetter	Plattform die eigene Größe verändert, einfügen von Triggern
10.09. - 16.09. MS02	Finalisieren der Bossmechaniken	Spawntruck, Überarbeitung vom Leben	Ramp Quest	Anordnung der Plattformen im Level, Rotierende und ZigZag Plattform
17.09. - 23.09. MS03	Bugfixing und letzte Abstimmung der Gegner, PP	Video	Bugfixing und Finalisierung	Springende Gullideckel, Boss-Plattform, PP

AUFGABENVERTEILUNG & AUFWAND

	Felix Schmidt	Marc Rosenkranz	Ardian Jahja	Josephine Eckhoff
25.06. - 01.07.	5	5	5	5
02.07. - 08.07.	0	0	0	0
09.07. - 15.07.	0	0	0	0
16.07. - 22.07.	0	0	0	0
23.07. - 29.07.	0	0	0	0
30.07. - 05.08.	8	17	25	10
06.08. - 12.08.	10	13	16	23
13.08. - 19.08.	7	9	18	13
20.08. - 26.08.	9	22	23	15
27.08. - 02.09. MS01	30	19	12	16
03.09. - 09.09.	25	20	22	27
10.09. - 16.09. MS02	23	14	29	9
17.09. - 23.09. MS03	12	26	10	15
Gesamt	129	145	160	133

CODEAUSSCHNITT JOSEPHINE

```
PlatformBreaks.cs
Assembly-CSharp
PlatformBreaks
Player

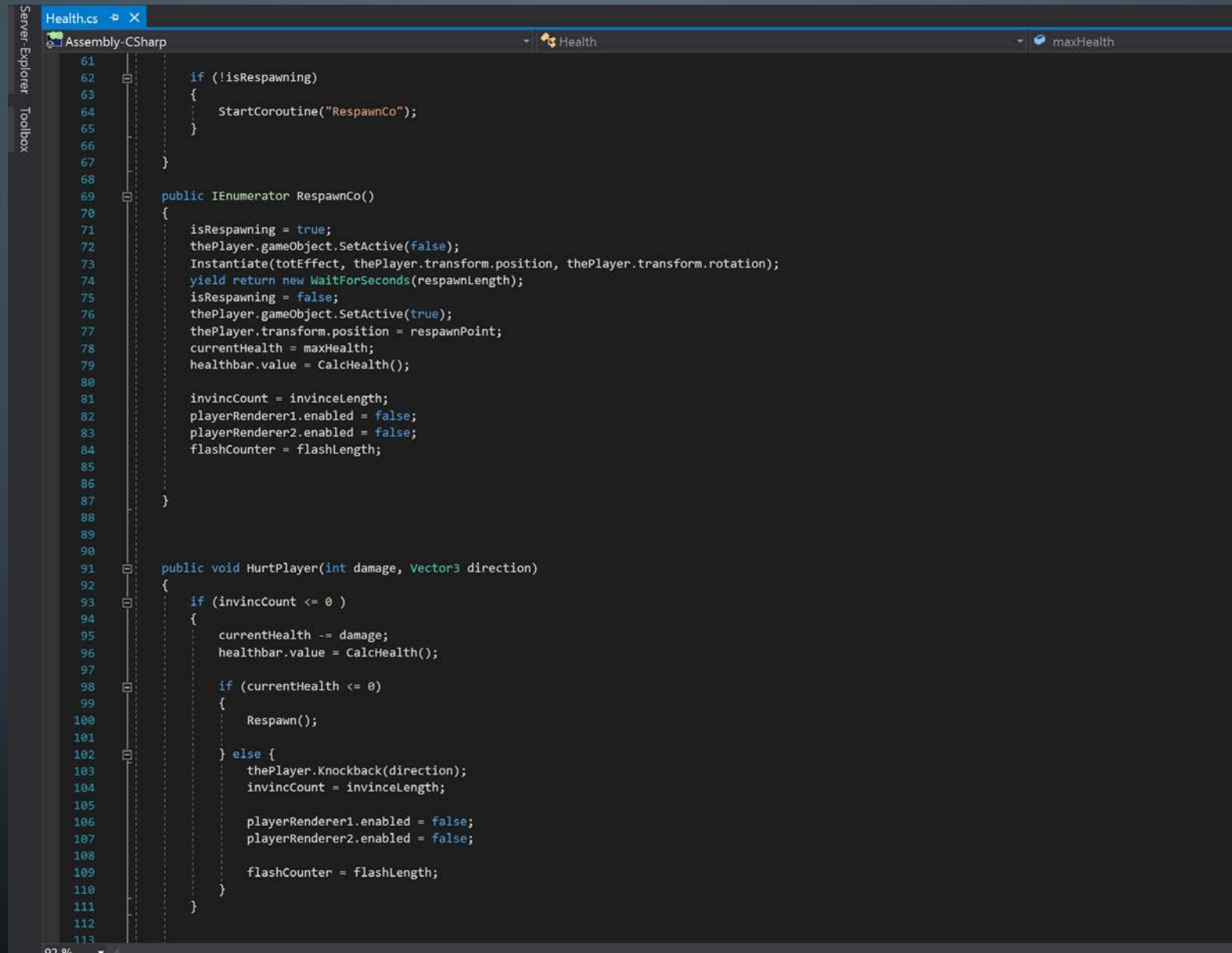
24 // Update is called once per frame
25 void Update () {
26
27
28 }
29
30 private void OnTriggerEnter(Collider other)
31 {
32     //BreakPlatform wird aufgerufen, wenn Player die Plattform berührt
33     if (other.gameObject == Player)
34     {
35         BreakPlatform();
36     }
37
38 }
39
40
41 public void BreakPlatform()
42 {
43
44     for (int x = 0; x < cubesInRow; x++)
45     {
46         for (int y = 0; y < 2; y++)
47         {
48             for (int z = 0; z < cubesInRow; z++)
49             {
50                 BrokenPieces(x, y, z);
51             }
52         }
53     }
54     gameObject.SetActive(false);
55 }
56
57
58
59 void BrokenPieces(int x, int y, int z)
60 {
61     //kleine Wuerfel erzeugen
62
63     GameObject smallCube;
64     smallCube = GameObject.CreatePrimitive(PrimitiveType.Cube);
65     smallCube.name = "cubePiece";
66
67     smallCube.transform.position = transform.position + new Vector3(cubeSize * x, cubeSize * y, cubeSize * z) - cubesPivot;
68     smallCube.transform.localScale = new Vector3(cubeSize, cubeSize, cubeSize);
69     smallCube.GetComponent<Renderer>().material = newMaterialRef;
70
71     smallCube.AddComponent<Rigidbody>();
72     smallCube.GetComponent<Rigidbody>().mass = 0.01f;
73     smallCube.GetComponent<Rigidbody>().AddForce(Random.Range(-scatter, scatter), 0, Random.Range(-scatter, scatter));
74     smallCube.GetComponent<Rigidbody>().AddTorque(Random.Range(-scatter, scatter) * 80, Random.Range(-scatter, scatter) * 80, Random.Range(-scatter, scatter) * 80);
75     Destroy(smallCube, duration);
76 }
```


CODEAUSSCHNITT ARDIAN

```
BezierCurve.cs
Assembly-CSharp
BezierCurve
Start

82 Vector3 oldPositionG = p0;
83 Vector3 newPositionG = Vector3.zero;
84
85 for (int i = 0; i < pointsNum; i++) {
86
87     float t = i / (float) pointsNum;
88
89     newPositionG = CalculateCubicBezierPoint(t);
90
91     DrawLines(cgreen, oldPositionG, newPositionG);
92
93     oldPositionG = newPositionG;
94
95 }
96
97
98 DrawLines(cblue, p0, p1);
99 DrawLines(cblue, p2, p3);
100
101 }
102
103
104 private void DrawLines(Color color, Vector3 p0, Vector3 p1) {
105
106     Gizmos.color = color;
107     Gizmos.DrawLine(p0, p1);
108
109 }
110
111 //Diese Funktion gibt einen Vector3-Punkt auf der Kurve mithilfe des Bezier-Algorithmus zurück
112 private Vector3 CalculateCubicBezierPoint(float t) {
113
114     return Mathf.Pow((1f - t), 3) * p0 + 3 * Mathf.Pow((1 - t), 2) * t * p1 + 3 * (1 - t) * Mathf.Pow(t, 2) * p2 + Mathf.Pow(t, 3) * p3;
115
116 }
117
118 void Update() {
119
120     if (triggerActivated) {
121
122         if (c < pointsNum) {
123
124             float t = c / (float) pointsNum;
125
126             newPosition = CalculateCubicBezierPoint(t);
127
128             ballbody.transform.Rotate(new Vector3(-100f*Time.deltaTime, 50f*Time.deltaTime, 0), Space.World);
129             ballbody.transform.position = newPosition;
130
131             oldPosition = newPosition;
132
133             c += Time.deltaTime*5f;
134
135         }
136     }
137 }
```

CODEAUSSCHNITT MARC



```
61
62 if (!isRespawning)
63 {
64     StartCoroutine("RespawnCo");
65 }
66
67 }
68
69 public IEnumerator RespawnCo()
70 {
71     isRespawning = true;
72     thePlayer.gameObject.SetActive(false);
73     Instantiate(totEffect, thePlayer.transform.position, thePlayer.transform.rotation);
74     yield return new WaitForSeconds(respawnLength);
75     isRespawning = false;
76     thePlayer.gameObject.SetActive(true);
77     thePlayer.transform.position = respawnPoint;
78     currentHealth = maxHealth;
79     healthbar.value = CalcHealth();
80
81     invincCount = invinceLength;
82     playerRenderer1.enabled = false;
83     playerRenderer2.enabled = false;
84     flashCounter = flashLength;
85
86 }
87
88
89
90
91 public void HurtPlayer(int damage, Vector3 direction)
92 {
93     if (invincCount <= 0 )
94     {
95         currentHealth -= damage;
96         healthbar.value = CalcHealth();
97
98         if (currentHealth <= 0)
99         {
100             Respawn();
101         }
102         else {
103             thePlayer.Knockback(direction);
104             invincCount = invinceLength;
105
106             playerRenderer1.enabled = false;
107             playerRenderer2.enabled = false;
108
109             flashCounter = flashLength;
110         }
111     }
112 }
113
```


CODEAUSSCHNITT FELIX

```

BossScript.cs* [X]
Assembly - CSharp
BossScript
controller

232     lastA = 2;
233     if (enemyInRange)
234     {
235         transform.position = Vector3.MoveTowards(transform.position, new Vector3(target.transform.position.x, transform.position.y, target.transform.position.z), 20f * Time.deltaTime);
236         transform.position = new Vector3(transform.position.x, transform.position.y, transform.position.z);
237         if ((Time.frameCount - 150) % 600 == 0){
238             attack = 0;
239         }
240     }
241     enemyInRange = false;
242     break;
243
244     case 3:
245         lastA = 3;
246
247         if (Time.frameCount % 100 == 0)
248         {
249             if (enemyInRange)
250             {
251                 targetPosi = transform.position - target.transform.position;
252                 xdis = targetPosi.x;
253                 zdis = targetPosi.z;
254             }
255         }
256
257         if (enemyInRange)
258         {
259             if (Time.frameCount % 100 >= 0 && Time.frameCount % 100 <= 24)
260             {
261                 transform.position = new Vector3(transform.position.x - (xdis / 50), transform.position.y + (maxHeight / 20), transform.position.z - (zdis / 50));
262             }
263             if (Time.frameCount % 100 >= 25 && Time.frameCount % 100 <= 49)
264             {
265                 transform.position = new Vector3(transform.position.x - (xdis / 50), transform.position.y - (maxHeight / 20), transform.position.z - (zdis / 50));
266             }
267             if (Time.frameCount % 100 == 50)
268             {
269                 enemyInRange = false;
270             }
271         }
272         enemyInRange = false;
273         break;
274
275     case 4:
276         lastA = 4;
277         if (enemyInRange)
278         {
279             if (Time.frameCount % 50 == 0)
280             {
281                 GameObject newBall = Instantiate(followBall, transform.position, transform.rotation) as GameObject;
282                 newBall.transform.position = new Vector3(transform.position.x, transform.position.y + 3, transform.position.z);
283                 newBall.AddComponent<DamagePlayer>();
284             }

```