Hochschule Aalen

Fakultät Optik und Mechatronik

Sommersemester 2023

Mini Game Fruchtschneider VR (VR Spiel)

Gruppe 11

VR/AR Dokumentation & Spielanleitung

Im Fach Autorensysteme

Lukas Gollhausen 83745

Jonathan Kechter 83701

Jonathan Möller 83443

Maximilian Zeger 82800

Hochschule Aalen

Fakultät Optik und Mechatronik

Sommersemester 2023

Mini Game Fruchtschneider VR (VR-Spiel)

Gruppe 11

VR/AR Dokumentation & Spielanleitung

Im Fach Autorensysteme

Lukas Gollhausen 83745

Jonathan Kechter 83701

Jonathan Möller 83443

Maximilian Zeger 82800

Betreuer:

Prof. Dr. Carsten Lecon

Stefan Wehrenberg

Aalen, den 12.07.2023

Eidesstattliche Erklärung

Hiermit erklären wir, dass wir die vorliegende Arbeit eigenständig und ohne fremde Hilfe angefertigt haben. Textpassagen, die wörtlich oder dem Sinn nach auf Publikationen oder Vorträgen anderer Autoren beruhen, sind als solche kenntlich gemacht.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Aalen, den 12.07.2023

Lukas Gollhausen

Jonathan Kechter

Maximilian Zeger

Jonathan Möller

Inhalt

Eidesstattliche Erklärung 5

1 Ideenfindung und Vorgehensweise 9

2 Verwendete Assets 11

3 Bestandteile des Spiels 12

3.1 Logik Spielverwaltung 12

3.2 Logik Katana-Schwert 15

4 Fazit 18

5 Spielanleitung 19

5.1 Spiel starten 19

5.2 Spielprinzip 19

5.3 Hauptmenü 20

5.4 Spielablauf 20

5.5 Spiel beenden 22

6 Quellen 23

1 Ideenfindung und Vorgehensweise

Da waren wir also, ein Team aus zwei User Experience Design-Studenten und zwei Informatik-Studenten. Wir hatten so gut wie kein Vorwissen/Vorkenntnisse über die Spielengine Unity. Wir wussten also erst einmal gar nicht was wir denn überhaupt für ein Spiel „erschaffen“ wollen, da wir Faktoren wie Zeit und Aufwand überhaupt nicht einschätzen konnten.

Also haben wir wie sonst auch uns hingesetzt und ein ganz klassisches Brainstorming durchgeführt. Wir hatten eine sehr breite Auswahl an Spielideen welche wir als Team ziemlich cool fanden. Doch schnell ist uns bei tieferer Recherche aufgefallen das es manchmal dann doch nicht so einfach ist, dass Gewünschte Spiel zu „erschaffen“, wie man sich das vorstellt. Zeit und Aufwand waren für eine Großzahl unserer Spielideen leider etwas zu hoch.

Wir haben also etwas umgedacht und haben nach relativ einfachen Spielprinzipien Ausschau gehalten, wie zum Beispiel Beat Saber-Spiele, Shooting Range-Spiele oder auch Spielen, wie Fruit Ninja bei den es nur darum geht, Früchte zu zerschneiden und dafür Punkte zu bekommen.

Schlussendlich ist dies dann auch das Spiel bzw. Spielprinzip geworden, an dem wir ansetzen wollten. Ein Spieler ist an einer festen Position und vor ihm werden Früchte hochgeschmissen welche er mithilfe eines Schwertes, in diesem Fall ein Katana-Schwert, zerschneiden kann und dadurch Punkte verdient. Natürlich soll das Spiel kein Endlosspiel sein, sondern die Herausforderung besteht darin, keine Bomben während dem zerschneiden zu erwischen da man sonst, nach drei zerschnittenen Bomben, verloren hat. Nach dieser groben Ideenfindung haben wir uns also herangesetzt und haben recherchiert, wie wir das gesamte Projekt angehen wollen und umsetzen können. Welche Ressourcen brauchen wir und wer aus dem Team kann welche

Fähigkeiten in das Projekt mit einfließen lassen um am Ende des Semesters ein erfolgreiches Mini VR-Spiel zu „erschaffen“. Wir haben also den verschiedenen Teammitgliedern die verschiedenen Aufgaben zugeteilt und schon konnte das Projekt starten.

2 Verwendete Assets

In diesem Projekt wurden mehrere verschiedene Assets benutzt, um das VR-Spiel Fruchtschneider VR aufzubauen. Die dazugehörigen Quellen sind jeweils im Quellenverzeichnis zu finden.

- Unity
- Blender
- GitHub
- Fruits
- Spielumgebung
- Katana-Schwert

3 Bestandteile des Spiels

In diesem Teil der Dokumentation soll es um die Logik des Spiels gehen. Um den Rahmen der Dokumentation nicht zu sprengen, werden wir uns hier nur auf die Logik der Spielverwaltung und des Katanaschwerts beschränken.

3.1 Logik Spielverwaltung

Im Spielverwaltung-Skript werden TextMeshPro-Variablen verwendet, um den Countdown, den Spielstand und den Highscore im Spiel anzuzeigen. Durch die Initialisierung der entsprechenden TextMeshPro-Komponenten können diese Texte in der Spielwelt angezeigt werden. Die TextMeshPro-Komponenten werden zu Beginn des Spiels initialisiert und in den entsprechenden Funktionen aktualisiert. Wenn der Spielstand oder der Highscore geändert werden, wird die Text-Eigenschaft der TextMeshPro-Komponenten entsprechend aktualisiert, um die neuen Werte anzuzeigen.

```
void Start () {
    countdownText = countdown.GetComponent<TMP_Text>();
    scoreText = score.GetComponent<TMP_Text>();
    highScoreText = hS.GetComponent<TMP_Text>();
    highScore = PlayerPrefs.GetInt("highscore");
    highScoreText.text = "Highscore:\n" + highScore.ToString();
}
```

Die Funktion "StartRound" wird aufgerufen, um das Spiel zu starten. Sie initialisiert die erforderlichen Variablen für den Spielbeginn, einschließlich der Zeit, des aktuellen Spielstands und des Countdowns. Dabei wird der Countdown-Text über TextMeshPro aktualisiert und die Funktionen "SpawnFruit" und "cd" mit InvokeRepeating aufgerufen, um Früchte zu erzeugen und den Countdown zu aktualisieren.

```
public void StartRound(){
    clappers.SetActive(false);
    time = 120;
    currentScore = 0;
    countdownText.text = time.ToString();
    scoreText.text = "Punktestand:\n" + currentScore.ToString();
    InvokeRepeating("SpawnFruit", 3.0f, 3.0f);
    InvokeRepeating("cd", 1.0f, 1.0f);
}
```

Die Funktion "ChangeScore" aktualisiert den Spielstand, abhängig von der übergebenen Änderung (change). Sie nutzt ebenfalls TextMeshPro, um den aktuellen Spielstand anzuzeigen.

```
public void ChangeScore(int change){
    if(time > 0){
        currentScore += change;
        scoreText.text = "Punktestand:\n" + currentScore.ToString();
    }
}
```

Die Funktion "cd" aktualisiert den Countdown und überprüft, ob die Zeit abgelaufen ist. Bei einem abgelaufenen Countdown wird der Highscore aktualisiert und in den PlayerPrefs gespeichert. Zudem werden bestimmte Aktionen wie das Anzeigen des Start- und Beenden-Buttons ausgelöst.

```
void cd(){
    time--;
    countdownText.text = time.ToString();
    if(time <= 3){
        CancelInvoke();
        InvokeRepeating("cd", 1.0f, 1.0f);
    }
    if(time == 0){
        CancelInvoke();
        if(currentScore > highScore){
            highScore = currentScore;
            PlayerPrefs.SetInt("highscore", highScore);
            highScoreText.text = "Highscore:\n" + highScore.ToString();
            clappers.SetActive(true);
            Animator[] anims = clappers.GetComponentsInChildren<Animator>();
            for(int i = 0; i < anims.Length; i++){
                anims[i].SetTrigger("clap");
            }
        }
        startFruit.SetActive(true);
        quitFruit.SetActive(true);
    }
}
```

Die Funktion "SpawnFruit" erzeugt Früchte in der Spielwelt. Dabei werden zufällige Positionen, Richtungen, Stärken und Rotationen verwendet, um eine natürliche und abwechslungsreiche Spielumgebung zu schaffen. Die Früchte werden nach einer bestimmten Zeit gelöscht, um die Rechenleistung zu optimieren.

```
void SpawnFruit(){
    for(int x = 0; x < 5; x++){
        GameObject randomFruit;
        if(Random.Range(0.0f, 100.0f) < 8.0f){
            randomFruit = bomb;
        }else{
            randomFruit = objectsToSpawn[Random.Range(0 , objectsToSpawn.Count)];
        }
        Vector3 pos = transform.position;
        pos.x += x;
        pos.z = pos.z - Mathf.Abs(2-x)*0.7f;
        GameObject newFruit = Instantiate(randomFruit, pos, randomFruit.transform.rotation);

        Vector3 randomVector = new Vector3(Random.Range(-0.1f, 0.1f), 1.0f, -0.06f);
        newFruit.GetComponent<Rigidbody>().AddForce(randomVector * Random.Range(7.5f, 9.5f), ForceMode.Impulse);
        newFruit.GetComponent<Rigidbody>().AddTorque(Random.Range(-0.1f, 0.1f), Random.Range(-0.1f, 0.1f), Random.Range(-0.1f, 0.1f), ForceMode.Impulse);
        Destroy(newFruit, 3.0f);
    }
}
```

3.2 Logik Katana-Schwert

Das Katanaschwert-Skript (Schwert.cs) ist für die Interaktion mit dem Schwert verantwortlich. Es ermöglicht das Zerteilen von Früchten und andere Aktionen, wenn das Schwert mit bestimmten Objekten kollidiert.

Im Katanaschwert-Skript werden verschiedene Funktionen verwendet, um die Logik des Schwerts umzusetzen. Wenn das Schwert mit einem Objekt kollidiert, wird die Funktion "OnTriggerEnter" aufgerufen.

Je nachdem, welchen Tag das kollidierte Objekt hat, werden unterschiedliche Aktionen ausgeführt. Wenn das Objekt den Tag "start" hat, wird die Funktion "StartRound" aus der Spielverwaltung aufgerufen, um das Spiel zu starten. Bei einem Objekt mit dem Tag "quit" wird das Spiel über "Application.Quit" beendet.

Für Objekte mit den Tags "fruit" oder "bomb" wird das EzySlice-Framework verwendet, um das Objekt in zwei Teile zu zerteilen. Die "Slice" Funktion von EzySlice wird aufgerufen, wobei das zu zerteilende Frucht-Objekt, die Schnittebene und eine optionale Materialzuweisung übergeben werden. Dieses Framework ermöglicht das präzise Zerteilen der Objekte entlang der definierten Schnittebene.

Nachdem das Objekt zerteilt wurde, werden die Ober- und Unterhälften des zerteilten Objekts erstellt. Für jede Hälfte wird die Funktion "AddComponents" aufgerufen, um die erforderlichen Komponenten hinzuzufügen. Zuerst wird ein BoxCollider hinzugefügt, um die Kollisionserkennung zu ermöglichen. Anschließend wird ein Rigidbody hinzugefügt, der eine physikalische Simulation der zerteilten Teile ermöglicht. Dabei wird die Eigenschaft "interpolation" des Rigidbody auf "Interpolate" gesetzt, um eine flüssige Bewegung zu gewährleisten. Zusätzlich wird eine Explosionkraft auf

den Rigidbody angewendet, um den zerteilten Teil von der Schwertbewegung wegzustoßen.

Die zerteilten Teile werden mit einer Lebensdauer von 3 Sekunden versehen, bevor sie mit "Destroy" gelöscht werden, um die Rechenleistung zu optimieren.Der Spielstand wird entsprechend aktualisiert, je nachdem, ob das zerteilte Objekt eine normale Frucht oder eine Bombe ist. Für eine normale Frucht wird der Punktestand um 10 erhöht, während für eine Bombe der Punktestand um -50 verringert wird.

```
void OnTriggerEnter(Collider other)
{
    if(!other.gameObject.CompareTag("bomb")) XRcontroller.SendHapticImpulse(0.5f, 0.2f);
    if(other.gameObject.CompareTag("bomb")) XRcontroller.SendHapticImpulse(1.0f, 1.0f);


    if (other.gameObject.CompareTag("start")){
        controller.GetComponent<Spielverwaltung>().StartRound();
        other.gameObject.SetActive(false);
        quitFruit.SetActive(false);
    }else if(other.gameObject.CompareTag("quit")){
        Application.Quit();
    }
    else if(other.gameObject.CompareTag("fruit") || other.gameObject.CompareTag("bomb")){
        GameObject fruitToSlice = other.gameObject;


        MeshRenderer fruitRenderer = fruitToSlice.GetComponent<MeshRenderer>();
        MeshFilter fruitFilter = fruitToSlice.GetComponent<MeshFilter>();
        Mesh fruitMesh = fruitFilter.mesh;


        Vector3 cuttingPlanePosition = transform.position;
        Vector3 cuttingPlaneNormal = transform.right;

        SlicedHull result = fruitToSlice.Slice(cuttingPlanePosition, cuttingPlaneNormal, null);
        Material currentFill = fruitFill;
        if(other.gameObject.CompareTag("bomb")) currentFill = bombFill;

        GameObject upperHullGameObject = result.CreateUpperHull(fruitToSlice, currentFill);
        GameObject lowerHullGameObject = result.CreateLowerHull(fruitToSlice, currentFill);
        AddComponents(upperHullGameObject);
        AddComponents(lowerHullGameObject);
        Destroy(fruitToSlice);
        int score = 10;
        if(other.gameObject.CompareTag("bomb")) score = -50;
        controller.GetComponent<Spielverwaltung>().ChangeScore(score);


    }
}
```

```
public void AddComponents(GameObject objPart)
{

    objPart.AddComponent<BoxCollider>();

    objPart.AddComponent<Rigidbody>().interpolation = RigidbodyInterpolation.Interpolate;
    objPart.GetComponent<Rigidbody>().AddExplosionForce(350, objPart.transform.position, 30);
    Destroy(objPart, 3.0f);
}
```

4 Fazit

Das Fach „Autorensysteme“ bot uns die tolle Möglichkeit, an einem spannenden Projekt im Bereich Augmented- und Virtual-Reality teilzunehmen. Wir als Team hatten eine Menge Spaß an dem gesamten Projekt und können viel neues Wissen und vor allem neue Fachkenntnisse mit in die Zukunft nehmen.

Es war sehr interessant mit Studenten eines anderen Studienbereichs zu arbeiten da man so an viel neues Wissen herankam. Sei es in Bezug auf Wissen über Designtechnische Aspekte oder Arbeitsweisen der Studenten im Bereich Informatik.

Neben den positiven Erfahrungen konnten wir auch ein paar negative Erfahrungen sammeln, welche uns sicher weiterbringen werden. Das gesamte Projekt war geprägt von Höhen und Tiefen da kein einziges Teammitglied Vorerfahrungen mit Unity hatte. Dadurch mussten wir uns auch immer weiter einschränken in dem was wir und als Ziel für das Projekt vorgenommen hatten. Doch schon nach kurzer Zeit sind wir in einen guten Workflow gekommen und konnten das Projekt trotz Höhen und Tiefen erfolgreich abschließen. Zwar mussten wir hier und da ein paar Abstriche machen aber wir als Team sind trotzdem sehr zufrieden mit dem Endergebnis und dem Wissen, welches wir aus diesem Projekt mit in die Zukunft nehmen dürfen.

5 Spielanleitung

5.1 Spiel starten

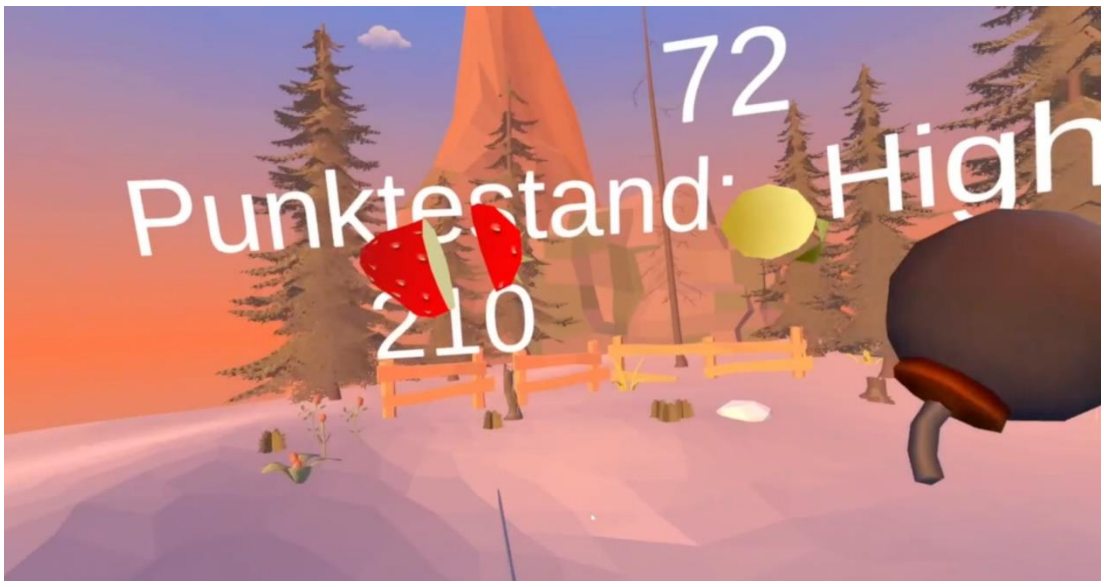
Um das Spiel „Fruchtschneider VR“ zu starten, muss die Datei „fruchtschneider.exe“ gestartet werden.

Hierbei sollte die VR-Brille schon mit dem entsprechenden Computer verbunden sein, um Komplikationen zu vermeiden.

5.2 Spielprinzip

Das Spiel orientiert sich an dem Smartphone-Spiel „Fruit Ninja“. Bei dem Spiel „Fruchtschneider VR“ geht es generell darum, in einer bestimmten Zeit (zwei Minuten) so viele Früchte wie möglich zu zerschneiden.

Durch das Zerschneiden von einzelnen Früchten erhält der Spieler pro normaler Frucht zehn Punkte. Allerdings können auch versehentlich Bomben zerschnitten werden, was zum Verlust von 50 Punkten führt.



Das Zerschneiden der Früchte passiert durch ein Katana, was an die rechte Hand des Spielers gebunden ist.

5.3 Hauptmenü

Nach dem Start des Spiels startet der Spieler in einer festen Umgebung. Vor ihm sind zwei Früchte zu sehen. Die eine Frucht (links) trägt die Schrift „Spiel starten“ über sich. Die zweite Frucht (rechts) trägt die Überschrift „Spiel beenden“.

Um das Spiel zu starten oder komplett zu beenden, muss der Spieler eine der beiden Früchte zerschneiden.

Wenn der Spieler zuvor das Spiel gespielt hat, sieht er hier auch seinen Score von dem vorherigen Spiel, sowie daneben den High Score.

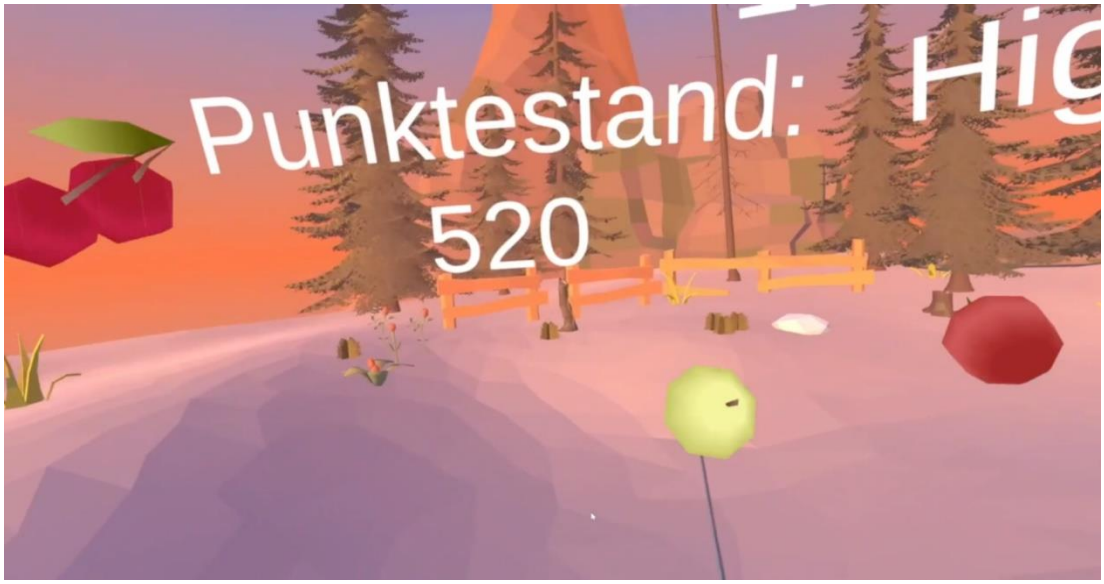
5.4 Spielablauf

Sobald der Spieler die Frucht „Spiel starten“ zerschneidet, verschwindet das Menü. Der Timer, der im Himmel zu sehen ist, erscheint und fängt an, zwei Minuten herunter zu zählen.

Simultan fängt es an, dass verschiedene Früchte (Apfel, Zitrone, Pfirsich, Erdbeere, Kirsche) in die Luft fliegen, welche zerschnitten werden müssen.

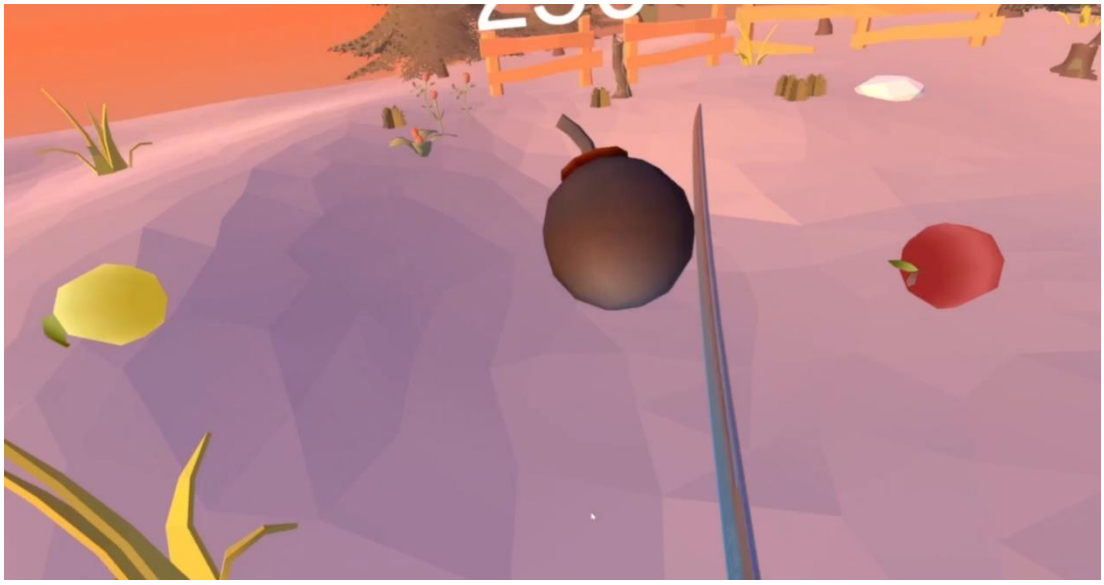
Diese Früchte ergeben pro zerschnittener Frucht 10 Punkte.

Die Früchte fliegen randomisiert durch die Luft und in verschiedener Reihenfolge.



Es werden allerdings nicht nur Früchte, sondern auch Bomben (schwarze Kugeln) zu randomisierten Zeitpunkten mit geworfen.

Wenn der Spieler versucht, diese zu zerschneiden, verliert der Spieler 50 Punkte.



Die gespielte Runde endet, wenn nach zwei Minuten der Timer ausläuft und der bis dahin erreichte Punktestand wird gespeichert.

Die erreichte Punktzahl wird dann weiterhin bis zum Starten einer neuen Runde oder bis zum Verlassen des Spiels angezeigt.

Neben der in der letzten Runde erspielten Punktzahl befindet sich zudem ein Highscore, der die höchste erspielte Punktzahl zeigt.

Das Spiel kann beliebig oft gestartet werden.

5.5 Spiel beenden

Wie vorhin erwähnt, gibt es eine „Menüfrucht“ bzw. eine Frucht, die durchschnitten werden kann, um das Spiel zu beenden.

Wenn dies passiert, schließt das Spiel komplett, ohne dass man dies noch einmal bestätigen muss.

6 Quellen

Unity https://unity.com/de

Blender https://www.blender.org/

GitHub https://github.com/

Fruits https://assetstore.unity.com/packages/3d/props/food/low-poly-fruit-pickups-98135

Spielumgebung https://assetstore.unity.com/packages/3d/environments/low-poly-free-vegetation-kit-176906

Katana-Schwert https://assetstore.unity.com/packages/3d/props/weapons/katana-vagabond-229627