# Spieleprogrammierung und 3D-Animation Praktikum 3

## Allgemeines

Im Praktikum zur Veranstaltung Spieleprogrammierung und 3D-Animation erlernen Sie den praktischen Umgang mit der Spiele-Engine Unity. Es dient als Ergänzung zu den theoretischen Inhalten aus der Vorlesung. Die grundlegenden Konzepte der Spieleentwicklung sind i. d. R. unabhängig von der genutzten Engine, so dass das Wissen, welches Sie bei der Entwicklung mit Unity erlangt haben, leicht auf andere Systeme übertragen werden kann. So sollte Ihnen neben dem Spezialwissen in Unity die Einarbeitung in andere Engines deutlich leichter fallen, wenn Sie das Praktikum erfolgreich absolviert haben.

Zum Bestehen des Praktikums benötigen Sie für das komplette Praktikum mindestens 80% der zu vergebenden regulären Punkte und dürfen kein Aufgabenblatt mit weniger als 50% der Punkte abschließen. Die Punkte pro Aufgabe erhalten Sie nur, wenn Sie beim Testat Ihre Lösungen schlüssig erläutern können.

Bei diesem Aufgabenblatt können Sie regulär 17 und mit Zusatzaufgaben 24 Punkte erreichen.

Das **Testat** zum Aufgabenblatt erfolgt am: **17.05. (Mi) & 26.05. (Fr)**

## Thema von Praktikum 3

Bei diesem Aufgabenblatt geht es darum, dass Sie Ihre erstellten Charakter-Animationen exportieren und diese in der interaktiven Umgebung von Unity steuern. Im Gegensatz zur Charakter-Steuerung aus Aufgabenblatt 1 soll bei dieser Praktikumsaufgabe die Steuerung Physik-basiert implementiert werden, so dass eine plausible Interaktion zwischen unterschiedlichen Objekten in der Szene abgebildet wird. Hierfür müssen Sie sich eingehend mit den Funktionen der Rigidbody-Komponente beschäftigen.

Didaktische Themen:

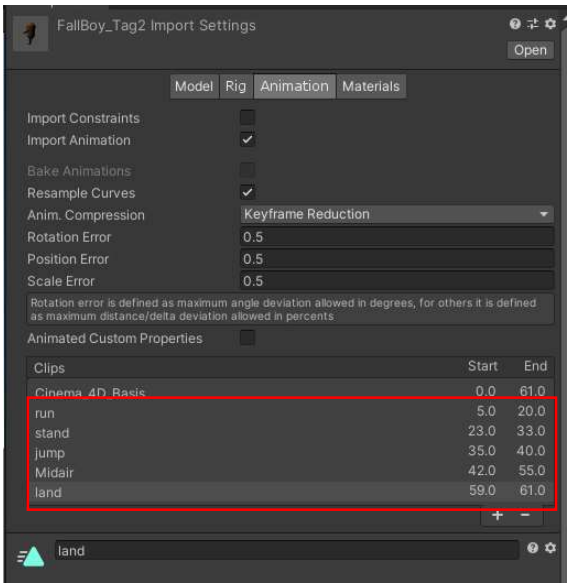
- Export und Import von FBX-Dateien
- Anlegen von State-Machines (hier Animation-Controller) zur Steuerung von Animationen in interaktiven Umgebungen
- Erzeugung eines eigenen Player-Controllers über Rigid Bodies
- Physikalische Interaktionen zwischen Rigid Bodies
- Ragdoll-Physics

## Aufgabe 1 (2 Punkte)

Exportieren Sie das Modell samt Animationen in das FBX-Format und prüfen Sie, ob diese korrekt in Unity dargestellt werden (entweder eine FBX-Datei mit einer Timeline, die alle Animationen hinteinander enthält oder aufgeteilt auf unterschiedliche FBX-Dateien (eine Animation+Modell pro FBX-Datei)).

Sofern Sie alle Animationen in einer Timeline exportiert haben, müssen Sie in Unity die Timeline in einzelne Animationen zerlegen. Hierfür müssen Sie das exportierte Modell-Asset auswählen und im

Reiter „Animation“ zusätzliche Clips mit den passenden Start- und Endframes versehen. Siehe folgendes Beispiel:




Wenn Sie jede Animation separat exportiert haben, entfällt dieser Schritt.

## Aufgabe 2 (4 Punkte)

Im ersten Praktikum wurde Ihnen ein Animation-Controller zur Verfügung gestellt, um die Animationen des Charakters in Ihrem Skript zu steuern. Damit Sie nun Ihren eigenen Charakter steuern können, müssen Sie auch für ihn einen Animation-Controller anlegen. Orientieren Sie sich dabei an dem vorgegebenen Animation-Controller aus Praktikum 1. Die Steuerung im Skript soll also über die Parameter „Grounded“ und „Speed“ erfolgen. Wenn Sie den BlendTree für den Movement-State anlegen, brauchen Sie nur die Gangarten hinzuzufügen, die Sie umgesetzt haben (vermutlich stehen und rennen, nicht rückwärts rennen oder gehen).

Fügen Sie Ihren neuen Charakter in die TipToeScene ein und tauschen Sie ihn gegen den Charakter aus Praktikum 1. Geben Sie ihm das Charakter Skript aus Praktikum 1, machen Sie ihn zum Ziel der Kamera und testen Sie, ob die Animationen alle richtig ablaufen. Haben Sie Ihren Animation-Controller richtig angelegt, müssen Sie nichts an Ihrem bestehenden Skript ändern.

Falls Sie eine zusätzliche Idle-Animation umgesetzt haben, sollten Sie Ihren Animation-Controller um einen entsprechenden Zustand erweitern und die Animation entweder automatisch abspielen lassen oder einen Auslöser dafür mit anlegen. Diesen könnten Sie dann über ein Skript aktivieren.

## Aufgabe 3 (4 Punkte)

Der wohl wichtigste Teil im Spiel Fall Guys ist, dass der Charakter auf seine Umgebung reagiert und von beweglichen Hindernissen umgeworfen und durch die Gegend geschleudert wird. Unity besitzt bereits ein integriertes Physiksystem, um genau solche Interaktionen zwischen GameObjects zu verarbeiten. In dieser Aufgabe sollen Sie die Steuerung des Charakters so implementieren, dass dieser korrekt mit dem Physiksystem interagiert.

Statt den Charakter direkt über die Transform-Komponente zu bewegen, bekommt er eine RigidBody-Komponente, die dann über Kräfte und Impulse gesteuert wird (z. B. AddForce(..)).

Erstellen Sie ein neues Skript „physicsCharacterControl“ und implementieren Sie in diesem die Steuerung, die auf Forces basiert. Orientieren Sie sich hier ruhig an Ihrem Skript aus Praktikum 1.

Ihr Charakter muss die gleiche Funktionalität haben wie in Praktikum 1, also über Maus und Tastatur gesteuert werden und fallen, wenn er keinen Boden unter den Füßen hat.

Implementieren Sie zusätzlich noch die Option zu **springen**.

Tipp: Wenn Sie Forces benutzen, um einen RigidBody zu verschieben und er trifft auf einen Collider, wird er entweder davor anhalten oder das getroffene Objekt verschieben, je nach Eigenschaften dieses anderen Objektes.

Tipp: In der RigidBody-Komponente gibt es eine Option für Schwerkraft. Wenn Sie diese aktivieren, wird Unitys interne Schwerkraft auf den RigidBody angewendet.

Tipp: Über Constraints können Sie bestimmte Achsen des RigidBodys sowohl für die Änderung der Rotation als auch für die Position blockieren.

Tipp: In manchen Fällen kann es sinnvoll sein, direkt die Geschwindigkeit (Velocity) des RigidBodys anzupassen.

## Aufgabe 4 (3 Punkte)

Nachdem Sie Ihren Charakter nun über Unitys Physiksystem bewegen, müssen Sie noch sicherstellen, dass seine Interaktion mit Hindernissen korrekt abläuft. In dieser Aufgabe müssen Sie ein passendes Verhalten Ihres Charakters für diese Interaktionen entwickeln. Folgende Ansprüche werden an die Verhaltensweise des Charakters gestellt:

Wenn der Charakter von einem anderen Physik-Objekt getroffen wird, muss er umfallen und weggeschleudert werden. Wie weit er fliegt und in welche Richtung ist dabei abhängig von der Geschwindigkeit und Richtung des Kollisionspartners.
Nach so einer Kollision muss der Spieler wieder normal bewegt werden können. Darum sollte er an der Stelle, an der er liegen geblieben ist, wieder aufgestellt werden. Das muss nicht animiert werden.

Testen Sie Ihre Implementierung zum Beispiel, indem Sie eine Kugel mit unterschiedlichen Geschwindigkeiten gegen den Charakter rollen lassen. Stellen Sie sicher, dass sich der Spieler nach einer Kollision genau so bewegt wie zuvor und dass auch danach weitere Kollisionen mit beweglichen Objekten stattfinden können. Legen Sie sich ruhig eine (oder mehrere) neue Szene(n) an, um mit der Verhaltensweise Ihres Charakters zu experimentieren.

Tipp: Sie können auch die Constraints eines RigidBodys über ein Skript ändern.

## Aufgabe 5 (4 Punkte)

Da der Spieler nicht nur auf rollende Kugeln treffen wird, sollen Sie weitere Physik-basierte Hindernisse anlegen. Implementieren Sie hierfür mindestens zwei der folgenden Objekte als Prefabs:

Wippe:
- Bewegt sich durch das Gewicht des Charakters runter.
- Ist sie zu steil, rutscht der Charakter ab und fällt.

Windmühle/Propeller:
- Rotierende Rotorblätter, die sich um den Mittelpunkt drehen.
- Schleudert den Charakter weg, wenn er getroffen wird. Wird davon nicht merklich gebremst.

Rotierende Scheibe:
- Dreht sich um den Mittelpunkt. Ein stehender Charakter wird einfach mit bewegt.
- Der Charakter kann darauf laufen und ist je nach Richtung schneller oder langsamer.

Fliegende Plattform:

Ähnelt der Scheibe. Kann sich in alle Himmelsrichtungen bewegen, aber nicht rotieren. Der Charakter kann darauf laufen und wird mit bewegt (wie im Fahrstuhl).

Rotierender Hammer:

Ein Hindernis, um den Charakter umzuwerfen. Dreht sich um einen Mittelpunkt. Wird durch eine Kollision nicht merklich gebremst.

Drehtür:

Eine Tür, die um einen Punkt dreht, die der Charakter aufschieben kann. Alternativ: eine automatische Drehtür, die den Charakter umwirft.

Pendel:

Eine Kugel oder Ähnliches an einem Seil oder einer Stange. Schwingt hin und her. Schleudert Charakter bei einer Kollision weg.

Stellen Sie sicher, dass der Charakter sich bei einer Kollision mit dem Hindernis richtig verhält und dass er wieder normal steuerbar ist, nachdem die Interaktion mit diesem Hindernis abgeschlossen wurde.

Tipp: Sie haben viele verschiedene Möglichkeiten, diese Aufgabe umzusetzen. Je nach Hindernis könnte es sinnvoll sein, wenn Sie sich Physics->Joints genauer angucken (Unity Joints, nicht zu verwechseln mit den Joints Ihres Charakter-Rigs)

## Extra Achievement 1 (4 Punkte)

Zurzeit verhält sich Ihr Charakter beim Umfallen wie eine Plastikfigur und bleibt starr. Viel schöner wäre es, wenn er einen klassischen Ragdoll-Effekt bekommen würde. Als Extra Achievement sollen Sie so einen Ragdoll-Effekt implementieren, der auftritt, wenn der Charakter umfällt. Vergessen Sie nicht, ihn beim Zurücksetzen auch wieder zu entfernen.

Tipp: Sie müssen den aktuellen Zustand der Animation in die Ragdoll-Rigidbodies überführen, damit die Ragdoll beim Aktivieren die richtige Haltung übernimmt. Damit auch die korrekten Geschwindigkeiten auf die Ragdoll-Rigidbodies übertragen werden, müssen Sie diese aus den vergangenen Positionen und Rotationen während der Animation berechnen (Differenz zwischen aktuellem Frame und dem davor). Wenn Sie die Differenz für die Rotationsgeschwindigkeit berechnen wollen, können Sie die Differenz-Quaternion durch $q_{diff} = q_i q_{(i-1)}^{-1}$ berechnen ($q_i$ entspricht der Rotation im aktuellen Frame und $q_{i-1}$ der Rotation im Frame davor).

## Extra Achievement 2 (3 Punkte)

Anders als in Praktikum 1 haben Sie in diesem Praktikum kein neues Minispiel angelegt. Legen Sie für diese Aufgabe ein neues Minispiel an, indem Sie einen Hindernisparcours in einer neuen Szene erschaffen, den der Charakter überwinden muss. Wie dieser Parcours genau aussieht, ist Ihnen überlassen und auch ob er weitere Umgebungsobjekte enthält.