Maturarbeit

Entwicklung eines Computerspiels mit Unity

# Einleitung

# Die Spiele-Engine Unity

## Die Unity Physik-Engine

### Das Dynamic Tree Modul

Ein Spiel besteht in der Regel aus sehr vielen Objekten. Für den Spielverlauf muss häufig die Beziehung zwischen Objekten geprüft werden, zum Beispiel zur Prüfung:

* ob zwei Objekte miteinander kollidieren,
* ob ein Objekt aus Perspektive der Spielfigur sichtbar ist,
* oder zum Finden des Bodens.

Dazu werden sogenannte Dynamic Trees und Raycasts verwendet.

<TODO> Graphik einfügen mit vielen Objekten und Raycast (siehe Folie 11)

Je mehr Objekte vorhanden sind desto länger würde die Prüfung dauern, wenn jedes Objekt einzeln berechnet wird (Brute Force Vorgehen). Um die Effizienz der Berechnung zu verbessern, werden Objekte daher in Gruppen zusammengefasst, wenn ein Raycast eine Gruppe trifft, wird die Berechnung nur innerhalb dieser Gruppe fortgeführt, wodurch insgesamt weniger Berechnungen benötigt werden.

<TODO> Graphik für Objektgruppen (Folie 18)

Die Struktur der Objektgruppen wird Bounding Volume Hierachy (BVH) genannt, zur Speicherung werden binäre Suchbäume verwendet.

<TODO> Graphik eines Bounding Volume Hierachy (Folie 18)

<TODO> Erklärung der Erstellung einer Bounding Volume Hierachy

Weil die Berechnung von Komplizierteren Formen langsam ist, werden die Objekte zuerst vereinfacht um diese Berechnung zu beschleunigen. Um jedes Objekt wird zuerst mit den oberen und unteren Ecken Axis Aligned Bounding Boxes (AABB) berechnet. Danach werden wiederum neue AABB erstellt, welche je zwei Objekte und deren AABBs enthalten, um die Menge an Möglichkeiten welche mit dem Raycast getroffen werden können zu reduzieren. Dieser Vorgang wird wiederholt bis nur noch eine grosse AABB übrig ist. Die AABBs werden in einem Binärbaum einer Bounding Volume Hierachy gespeichert.

<TODO> Erklärung des Such-Algorithmus

Mit einem Raycast wird nach dem nächsten Objekt auf einer Linie gesucht. Die einfachste Methode ist jedes Objekt einzeln zu überprüfen, aber diese Methode ist sehr ineffizient. Durch die davor erstellte BHV können zuerst die grössten zwei AABBs kontrolliert werden und somit werden von Anfang an viele Objekte ausgeschlossen. Dieser Vorgang kann wiederholt werden bis die AABBs nur noch die zwei Objekte enthalten. Weil dadurch mit jedem vergleich die hälfte der Möglichen Objekte ausgeschlossen wird ist es viel effizienter als das Brute Force vorgehen.

<TODO> Die drei Vorgehensvarianten (Bottom Up, Top Down, Incremental)

Bei dem Bottom Up Vorgehen werden die Objekte mit dem nächsten anderen Objekt in eine AABB gemacht. Danach wird dieser Vorgang mit den neu erstellten AABBs wiederholt. Bis nur nuch eine AABB übrig ist. Dadurch werden alle Objekte gleichmässig in der BVH verteilt.

Während dem top down verfahren wird zuerst eine grosse AABB erstellt welche alle Objekte beinhaltet. Danach werden die Objekte in zwei Gruppen geteilt. Dafür gibt es zwei Möglichkeiten, Die AABB wird entweder in der Mitte in zwei Gruppen aufgeteilt oder es werden die AABBs mit der geringsten Oberfläche gesucht. Bei der ersten Variante kann die Unterteilung durch die einfachere Berechnung effizienter durchgeführt werden aber die Gruppen sind zufällig. Bei der zweiten Variante wird die geringste Oberfläche der AABBs gesucht. Dadurch dauert das Erstellen des BVH am Anfang Länger aber Weil die Wahrscheinlichkeit das ein Raycast eine AABB trifft, proportional mit der Oberfläche steigt treffen weniger Raycasts welche eigentlich kein Objekt treffen würden die AABB. Diese AABBs werden wiederum so lange unterteilt, bis jedes AABB nur noch zwei Objekte enthält.

Das incremental Verfahren fängt mit einem einzelnen Objekt im BVH an und es werden die weiteren Objekte eingefügt und mit einem anderen Objekt in eine AABB zusammengefasst. Der dadurch entstehende Baum ist meistens ineffizient und kann durch einen Algorithmus verbessert werden.

<TODO> Der Algorithmus zum Einfügen weiterer Objekte

Wenn ein neues Objekt in die BVH eingefügt werden soll, wird mit einem der anderen Objekte eine neue Node mit AABB gebildet welche die beiden Objekte enthält. Die darüberliegenden AABBS werden so angepasst das sie da das neue Objekt einschliessen. Welche der Objekte die beste Möglichkeit ist, wird durch die Surface Area Heuristic mithilfe der Einfüge Kosten berechnet.

<TODO> Die Surface Area Heuristic (SAH) zur Berechnung der Einfüge Kosten

Die Einfüge Kosten für jede node wird mit der Oberfläche der AABBs berechnet. Weil es sehr ineffizient ist, jede einzelne einfüge Möglichkeit zu überprüfen wird die Surface Area Heuristic (SAH) verwendet. Dazu wird die Fläche der neu hinzugefügten Node und die Anpassungen der darüberliegenden Nodes addiert.  
<TODO> Bsp Formel mit Bild   
Es werden die Kosten aller Nodes auf einer ebene Ausgerechnet und wenn die neu angepasste Node besser als die davor Beste ist wird sie als neue beste Node eingetragen. Dieses vorgehen wird bei den ???? der besten Node der Ebene wiederholt. Wenn die beste Node bestimmt ist wird kontrolliert, ob es überhaupt möglich ist das die darunterliegenden Nodes effizienter sind. Dafür wird überprüft, ob die Flächenänderung der darüberliegenden AABBs plus das AABB des neuen Objekts weniger kosten als die aktuellen besten Kosten.  
<TODO>grafische Erklärung und Formel

# Die Entwicklung von «Das Spiel»

# «Das Spiel»

# Nachwort

# Quellenverzeichnis