CT Projekt: Boids

Schwarmsimulation mit SFML Graphic Library in C++

Gabriel Kraus und Alexander Klee

20.10.2021 - 11.01.2022

Contents

1 Projektbeschreibung

In der Projektplanungsphase haben wir uns dazu entschieden “Boids” zu simulieren. Nach einiger Überlegung entschieden wir uns für die Programmiersprache C++. Diese ist für Graphikanwendungen zwar eher weniger bekannt und schwieriger zu nutzen, jedoch ist sie sehr gut geeignet für aufwändige berechnungen. Das kann wichtig werden, wenn man eine sehr große Anzahl an Boid-Objekten Simuliert, da die Rechenzeit pro Boid exponentiell steigt.
Boids sind eine Simulation von künstlichem Leben und folgen 3 Regeln:

- Cohesion - Zusammenhang, von Boids
- Separation - Separation, von anderen Boids
- Alignment - Angleichung, der Bewegungsrichtung der anderen Boids

Dabei entsteht eine Art Muster aus dem Chaos, der zufällig erzeugten Boids. Sie bilden kleine Gruppen und schwärmen umher. Die Technologie der Boids findet Anwendung in Filmen und Videospiele z.B. Batman Returns, bei dem ein großer Schwarm von Fledermäusen simuliert wird oder Half-Life bei dem Vogelartige Kreaturen dargestellt werden.

2 Ziele

2.1 Mindest Ziele

Das Ziel des Projektes war eine Simulation der Boids flüssig mit mindestens 100 Objekten zu simulieren, welche sich nach den drei Regeln Separation, Cohesion und Alignment gegenseitig beeinflussen. Zudem war ebenfalls die Idee anhand des Projektes C++ zu lernen und Erfahrung mit der Grafik-Bibliothek SFML sammeln.

2.2 Weiter Ziele

Wenn noch genügend Zeit verbleibt sind weiter Ziele:

- hinzufügen von neuen Boids durch Maus-Inputs
- anpassung der drei Parameter Separation, Cohesion und Alignment, sowie der Viewrange der Boids
- ‘Feindliche‘ Boids, welche den Rest der Boids jagt
- Räumliche Daten Struktur, wie einen Quad Tree, mit welchem die exponentielle Natur der Boids gebändigt werden kann

3 Ergebnisse

Dadurch, dass Wir (vor allem Alex) noch C++ lernen mussten und keinerlei Erfahrung mit SFML hatten, ist sehr viel Zeit dabei verbracht worden ein erstes laufendes Beispiel mit SFML zu erstellen. Vor allem das Aufteilen des Programms auf mehrere .cpp Dateien hat leider nicht mit dem Compiler Mingw-32 funktioniert. Um dies trotzdem zu ermöglichen haben wir dann mitten im Projekt die Migration zu Visual Studios, welches automatisch das Makefile richtig erstellt, begonnen.

Gelungen ist ein Boid-Simulations-Programm, welches anhand von drei anpassbaren Parametern problemlos hunderte, je nach PC sogar tausend Boids und deren Interaktionen Simulieren kann. Durch drücken der linken Maustaste werden weitere Boids hinzugefügt, und simuliert. Des weiteren ist es möglich die drei Parameter Separation, Cohesion und Alignment während das Programm läuft zu ändern, wobei nach Aktuellem stand das User-Interface, zwar wahrscheinlich bei Minimalisten gut ankommt, für normale Benutzer jedoch weniger als optimal ist.
Zeit für feindliche Boids oder einer ausgeklügelten Datenstruktur für die Position der Boids hatten wir jedoch leider nicht mehr.

4 Zusammenarbeit

Leider ist es schwierig die genaue Leistung der einzelnen Personen im Nachhinein zu ermitteln, auch deshalb, da ein signifikanter teil der Zeit damit verbracht wurde C++ genauer kennen zu lernen und SFML zum Laufen zu bekommen und sich mit dessen Dokumentation auseinander zu setzten. Das Projekt hat ungefähr eine gesamt Arbeitszeit von 27.5 Stunden, inklusive der Zeit im Unterricht, der Zeit zu Hause alleine und der Zeit in der wir uns im privaten Raum getroffen haben.

5 Set-Up von SFML

Von unserer Erfahrung empfehlen wir die Anleitung von sfml-dev.org, der offiziellen Website der Graphikbibliothek. Die Anleitung ist für Visual Studio Versionen 2017 und älter, funktioniert jedoch auch problemlos mit neueren Versionen. Es ist auch möglich sich selbst SFML mit CMake zu 'compilieren' und dann mit einem Compiler manuell alles zu linken, jedoch sollte man hierbei sich schon relativ gut mit dem Linker und Compiler auseinander gesetzt haben.

6 Programm

6.1 Fähigkeiten

Nach dem Starten des Programms fängt direkt die Simulation an, werden so viele Boids auf dem Bildschirm zufällig verteilt wie in main() angegeben wurden. Jedem dieser Boids wird auch ein zufälliger Gechwindigkeitsvektor zugewiesen, welcher sich jeden Frame nach den drei Regeln neu berechnet. Wenn die Linke Maustaste gedrückt gehalten wird, werden kontinuierlich neue Boids an der Position der Maus erstellt. Mit der Maus lässt sich ebenfalls mithilfe der drei Rechtecke oben links die drei Parameter Separation, Cohesion und Alignment anpassen. Wenn auf das am weitesten links gelegene Rechteck die Rechte Maustaste drückt, nimmt die Separation zu, mit der Linken Taste ab. Die mittlere Box Kontrolliert den Wert Alignment, und die rechte Box die Cohesion.

6.2 Funktionsweise

Die einzelnen Boids bewegen sich nach drei Regeln: Separation, Cohesion und Alignment. Jeden Frame werden diese neu Berechnet.
Die Methode Boid::update_movement() Berechnet für jeden dieser Regeln neuen Vektoren und addiert diesen zu dem Geschwindigkeitsvektor (Velocity).

Somit wird jeden Frame für jeden Boid folgendes berechnet:

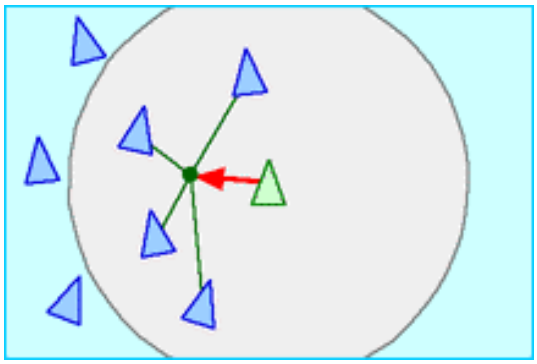


Figure 1: Cohesion Bei der Cohesion wird die Positionen der Boids im Sichtradius summiert und danach durch die Anzahl der Boids geteilt. Davon wird dann die Position des aktuellen Boids abgezogen um die Distanz zu diesem Mittelpunkt zu bekommen.

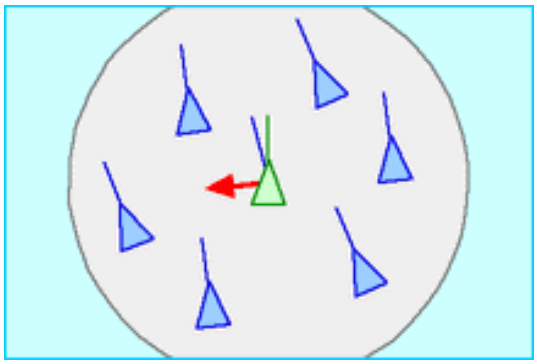


Figure 2: Alignment Das Alignment ist sehr ähnlich zur Cohesion, allerdings wird die Geschwindigkeit summiert und dann dividiert, um den Durchschnitt der Geschwindigkeit zu bekommen.

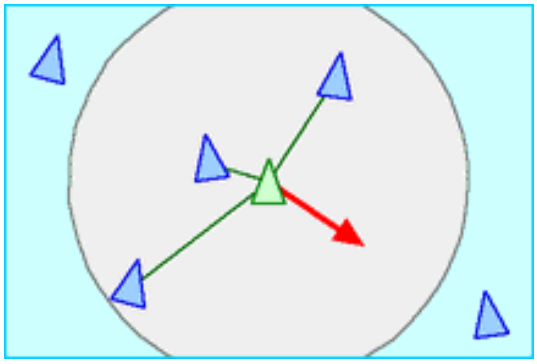


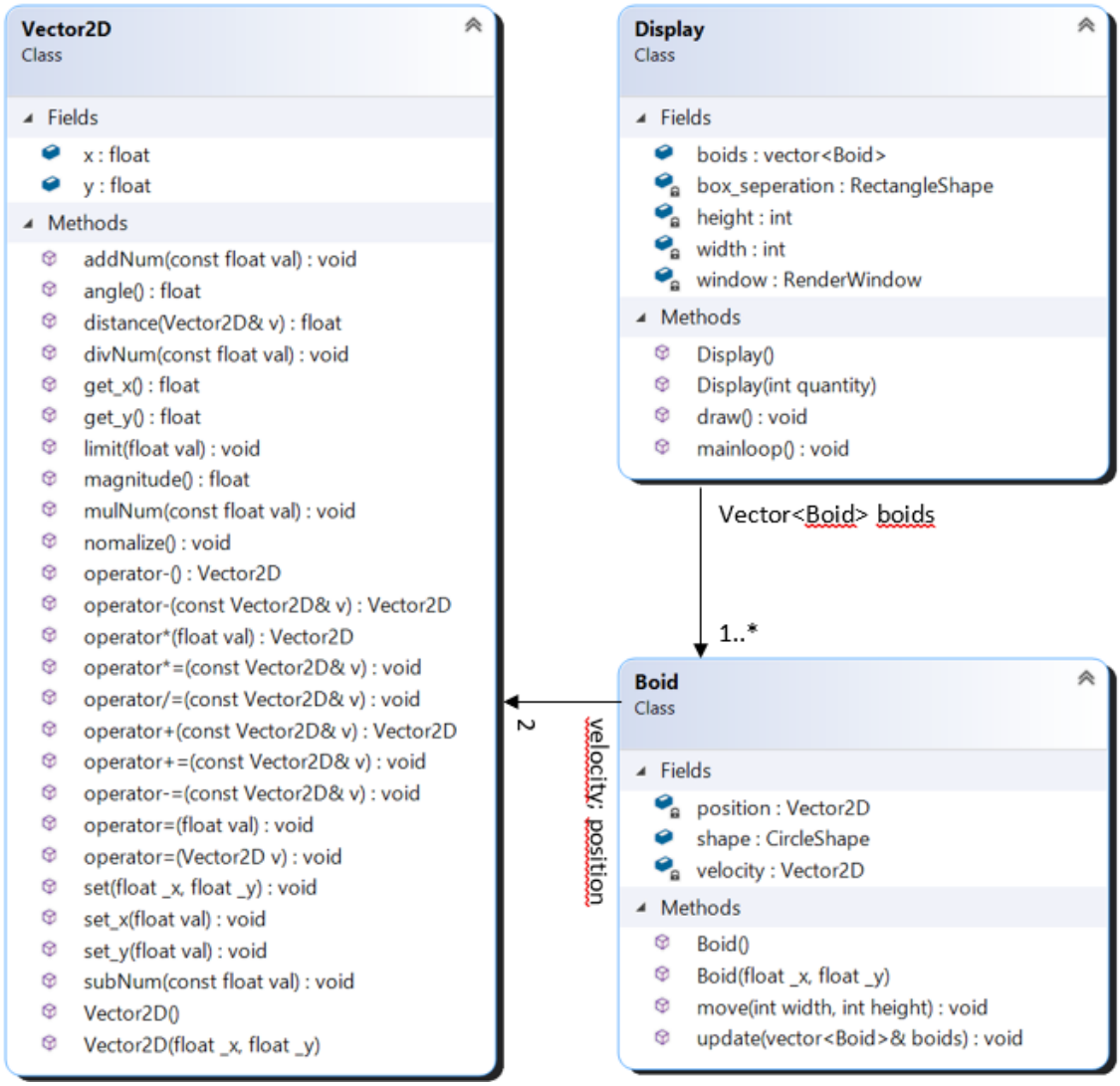
Figure 3: Separation Genauso wie bei Alignment und Cohesion wird bei der Separation ebenfalls ein Vektor aufsummiert, jedoch wird dieses Mal die Distanz summiert und durch die Anzahl dividiert. Dieser Durchschnitt wird später negiert (umgedreht), weil die Separation von den Boids wegzeigen soll, nicht dahin.

6.3 Bugs

Während dem Programmieren sind natürlich unzählig viele Bugs aufgetreten, jedoch waren ein paar dieser Bugs besonders interessant oder haben viel Zeit gekostet. Ein Beispiel für einen Bug der sogar beides ist, ist ein 'include loop' gewesen. Dadurch, dass die Header-Files in einer Falschen Reihenfolge waren, konnte der Compiler nie zur Definition von Boid und Vector2D kommen. Ein Temporärer fix war es eine Deklaration (Bsp.: class Boid;) ohne Definition von den beiden Klassen in Display.hpp und Boid.hpp zu machen. Inzwischen sind die Header jedoch in der richtigen Reihenfolge, wodurch dies sowieso kein Problem mehr ist. Die am Bugs welche am frustrierendsten waren jedoch die Bugs, welch nicht mal einen Compiler Fehler erzeugt haben. Das kam öfters vor während des implementieren der drei Regeln nach welchen sich die Boids bewegen. Ein Beispiel dafür ist das lange Zeit dieser Algorithmus einfach nicht wirklich wollte - nach langwierigen Debuggen und Loggen von verschiedensten Variablen unter all möglichen Zuständen, stellte sich heraus das die Funktion zum Berechnen der Distanz kompletter Mist war. Wo diese wirklich herkam ist uns immer noch unklar, eine Vermutung ist, dass sie nur halb hingeschrieben und danach einfach vergessen wurde. Nachdem diese Funktion gefixt wurde funktionierte der Algorithmus ohne Probleme.

6.4 UML diagramm

Das UML-Diagramm beinhaltet nur die Klassen die von uns geschrieben worden sind, keine Assoziationen zu Klassen von SFML oder C++.



7 Quellen

- http://www.red3d.com/cwr/boids/
- https://www.sfml-dev.org/
- https://stackoverflow.com/
- https://en.cppreference.com/w/
- https://www.geeksforgeeks.org/c-plus-plus/
- https://en.wikipedia.org/wiki/Boids
- https://eater.net/boids
- https://cs.stanford.edu/people/eroberts/courses/soco/projects/2008-09/modeling-natural-systems/boids.html
- https://github.com/jyanar/Boids/tree/master/
- https://www.reddit.com/r/sfml/
- https://www.youtube.com/watch?v=QbUPfMXXQIY
- https://gamedevelopment.tutsplus.com/tutorials/3-simple-rules-of-flocking-behaviors-alignment-cohesion-and-Separation–gamedev-3444
- https://www.youtube.com/watch?v=18c3MTX0PK0 &list=PLlrATfBNZ98dudnM48yfGUldqGD0S4FFb
- https://www.youtube.com/watch?v=tYspMwzV8w