Dokumentation für Mutiplayer-Autorennspiel “Drift”

Informatik Leistungskurs 2

Zai Shi

Geliederung

1. Einleitung
2. Physik und Bewegung
3. Netzwerk und Sycronisierung
4. Graphik und Modell
5. UI und Interaktion
6. Fazit
7. Quellen
8. Einleitung

Das Spiel “Drift” ist ein Autorennspiel für mehrere Spieler in einem LAN. Das Spiel ist mit der Unity-Engine [1] programmiert. Das Spiel läuft auf Android und Windows. Die Modelle wie die Häuser und Autos sind aus dem Unity-Assetsstore [12].

Ich mag Vogelperspektive Autorennspiel mit Drift sehr gerne, wie Drift-Runner2 [20] oder Smash-Cops-Heat [21]. Daswegen habe diese Idee meine eigenes Autorennspiel mit Unity zu machen.

1. Physik

Die Rechnungen der Physik des Autos sind mit dreidimensionale Vectoren(Vector3 [2]) in Script CarController. Ein Auto hat immer einige physikalische Attribute:

Masse (float mass)

Bewegungsrichtung und Geschwindigkeit (Vector3 velocity)

Position (Vector3 position)

Forwarts Richtung (Vector3 forward)

Aufwarts Richtung (Vector3 upward)

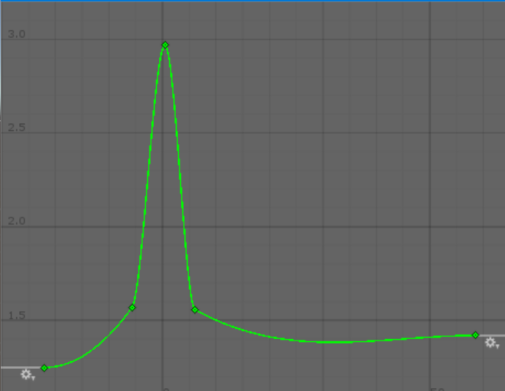
Ein Auto hat auch einige Attribute für die Bewegung (Ein AnimationCurve [3] ist eine Kurve, die man direkt in der Graphik bearbeiten kann):

Beschleunigung/rückwarts Beschleunigung (float acceleration/backAcceleration)

Trägheit der Beschleunigung (AnimationCurve accelerationDrag)

Beschleunigung der Drehung (AnimationCurve torque)

Trägheit der Rotation(float rotateDrage)

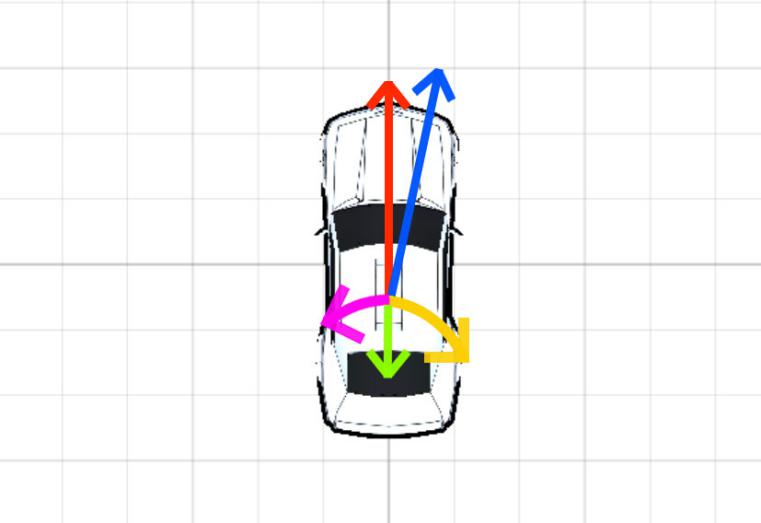
accelerationDrag und torque

Die Rechnung der Bewegung:

velocity += InputY() \* forward \* acceleration \* deltaTime (deltaTime ist die Zeit zwischen jedem Frame, InputX()/InputY() ist der horizontale/vertikale Input des Spieler zwischen -1 und 1)

AddTorque (InputX() \* upward \* torque.Evaluate(velocity.magnitude) \* deltaTime)

(AddTorque() ist ein Funktion, die eine Beschleunigung zu der Rotation des Autos hinzufügt. Evaluate() nimmt den Wert der X-Achse und gibt den Wert der Y-Achse der Kurve zurück. magnitude ist die Länge eines Vectors)



Rot: anfangs velocity, Grün: accelerationDrag, Gelb: torque, Lila: torqueDrag, Blau: neue velocity

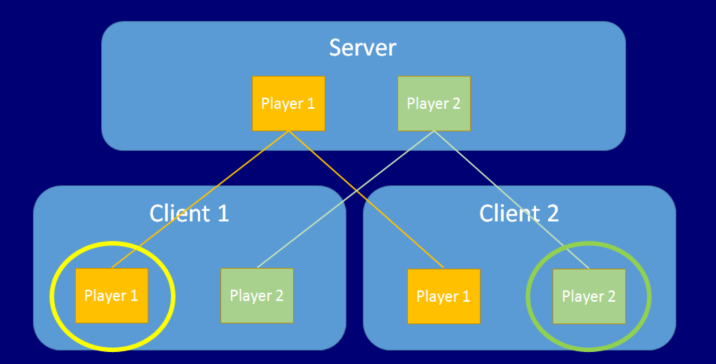
Nach jedem Frame berechnen wir die Reibung:

Velocity \*= (1-accelerationDrag.Evaluate(velocity.z)) \* deltaTime

angularVelocity \*= (1-rotateDrag) \* deltaTime

1. Netzwerk und Syncronisierung

Ich habe die Netzwerk-Module Unet für meine Projekt benutzt. Unet ist ein high-level-scripting-API [5]. In Unet ist ein Client von alle Spieler auch als ein Server. Unet controliert den Netzwerk mit eine NetworkManager [5]. Unet kann auch die Informationen und Befehle zwischen die Spieler schicken.



*Server-Client Diagramm für UNet. Die Spieler in der Kreise sind die Spiel von dem Client krontroliert wird[6]*

Wenn die Spieler anfangen in ein LAN zu spielen, die wissen nicht die IP-Addresse von der andere Spieler, daswegen brauchen wir eine Script, die IP zu alle Port in diese Lan broadcastet, und die IP von dem Anderen zu empfangen.

In die Hauptmenü (MainMenu) ist ein Script heißt ConnectionManager, welche NetworkDiscovery [7] erbt. Der hat vier Funktionen: StartServer(), StartClient(), CloseConnection() und überschriebende Funktion OnReceivedBroadcast(). Die erste drei Funktionnen können ein Server oder Client für die IP-Broadcast erstellen und schließen, die letzte Funktion kann die IP Adresse empfangen und behandeln.

In die haupte Szene (Town) habe ich zwei Scripts für die Netzwerk, eine ist NetworkManager. Die andere ist AutoConnection, welche mit dem API von NetworkManager zu mit dem IP, die wir vorher in Hauptmenü empfangt, der Server verbindet, wenn der Spieler als Client ist, oder ein Server erstellt, wenn der Spieler als Server ist.

Die CarController erbt NetworkBehavior [8]. die kontroliert nicht nur die Physik der Spieler auch die Sycronisierung der Positionen und Bewegungen. Es hat zwei Funktion heißt RpcSyncPos() mit Attribute [ClientRpc][9] und CmdSendServer() mit Attribute [Command][10], CmdSendServer() schickt die Position, Rotation, Geschwindigkeit und ID der Spieler von einem Client zu dem Server, und auf dem Server wird RpcSyncPos() aufgerufen und die Informationen zu die andere Client weiter schicken. Wenn die Andere Client die Informationen bekommen, dann wird LerpPosition() aufgerufen, die die Position der Spieler mit Lerp() [11] flüssig auf die neue Position stellen.



*Zwei Spieler in einer Szene*

1. Graphik und Modell

Die Modelle sind alle kostenlose Materiale vom AssetStore.

Die Automodelle [13] sind von “Modern Sports Car”, Ich habe die Texturen einbisschen verändert, um die Umgebung der Spiel besser passt, ein Lmape auf dem Polizeiauto und Bewegung der Reifen gemacht.



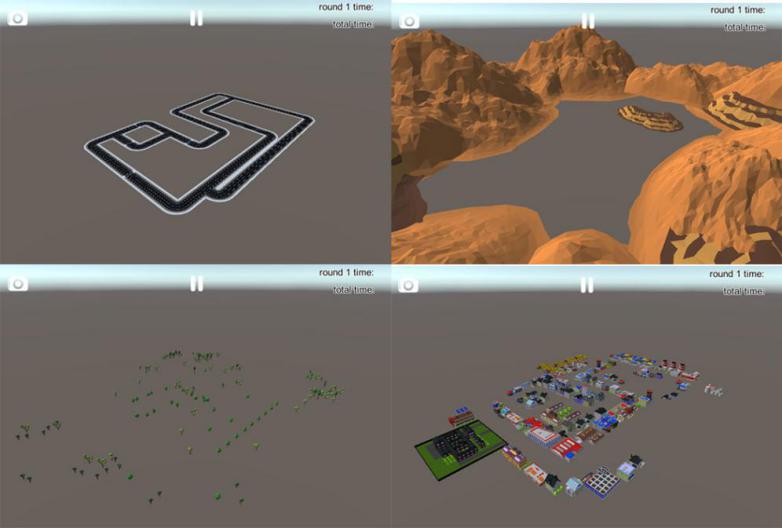
*Automodelle*

Die Straße-Asset ist von “Lowpoly Modern City Decorations Set” [14]. Die Bergemodelle kommt aus “LowPoly Environment Pack” [15]. Die Bäume sind aus “LowPoly Environment Pack”, “Lowpoly Modern City Decorations Set” und “LowPoly Vegetation Season Pack Lite” [16]. Die Häuser kommen aus “Lowpoly Modern City Buildings Set” [17]



*Straße, Berge, Bäume und Gebäude in Einzelteile*

Ich habe die Asset für meine Projekt in richtige größe und Positionen eingestellt, und ein Kleine Stadt bebaut.



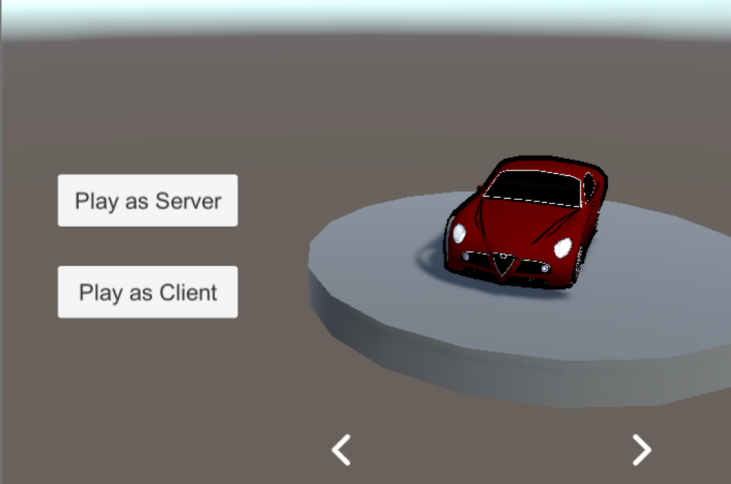
*Straße, Berge, Bäume und Gebäude in dem Szene*



*Überblick von die Hauptszene (Town)*

1. UI und Interaktion

In Hauptmenü kann man die Autos shauen, drehen und auswählen.



*Hauptmenü (MainMenu)*

Wenn man auf Play klicken, kommt man aus nächste Seite, wo man die andere Spieler warten. Wenn die Client dem Server schon verbindet, zeigt die Client auf dem Bildschirm der IP des Servers. Danach könne die Spieler auf Ready klicken und Spiel anfang zu spielen.



*UI für Warten und Verbinden*

Es gibt zwei Perspektive im Spiel, eine von oben, und eine von hinten. Man kann die wechseln mit dem linksobende Knopfe. Auf Windows-Desktop kontroliert man das Spiel mit WASD-Taste oder ↑←↓→-Taste. Auf dem Android-Handy biegen das Auto mit dem Bewegung des Handy (Schwerkraft-Sensor Gyro() [18]) ab, und die linke Seite des Bildschirm ist beschleunigen, rechte Seite ist bremmsen un rückwerts fahren.

*Unterschiedliche Perspektive*

1. Fazit

Es gibt noch viele Sachen in das Spiel, die man noch verbessern kann. Das Ziel des Spiel ist wegen zeitliche Grund nicht fertig programmiert. Das spiel soll nach 3 oder 5 Runde den Ziel erreichen, und zeigen, welche Spieler der schnellste ist. Die Graphik kann auch viel verbessert werden, die Reflektion des Texturen, das Licht (Global Illumination [19]) und die Liestung. Das Spiel hat jetzt nur 60 FPS auf Core-M mit 1080x1920 Auflösung.

Das Projekt ist jetzt auf meine eigenes Github (<https://github.com/FomalhautB/Drift>)

1. Quellen:
2. <https://unity3d.com>
3. <https://docs.unity3d.com/ScriptReference/Vector3.html>
4. <https://docs.unity3d.com/ScriptReference/AnimationCurve.html>
5. <https://docs.unity3d.com/Manual/UNet.html>
6. <https://docs.unity3d.com/ScriptReference/Networking.NetworkManager.html>
7. <https://docs.unity3d.com/uploads/Main/NetworkLocalPlayers.png>
8. <https://docs.unity3d.com/ScriptReference/Networking.NetworkDiscovery.html>
9. <https://docs.unity3d.com/ScriptReference/Networking.NetworkBehaviour.html>
10. <https://docs.unity3d.com/ScriptReference/Networking.ClientRpcAttribute.html>
11. <https://docs.unity3d.com/ScriptReference/Networking.CommandAttribute.html>
12. <https://docs.unity3d.com/ScriptReference/Vector3.Lerp.html> <https://docs.unity3d.com/ScriptReference/Quaternion.Lerp.html>
13. <https://assetstore.unity.com/>
14. <https://assetstore.unity.com/packages/3d/vehicles/land/modern-sports-car-8-2209> <https://assetstore.unity.com/packages/3d/vehicles/land/modern-sports-car-5-2203> <https://assetstore.unity.com/packages/3d/vehicles/land/modern-sports-car-10-5368> <https://assetstore.unity.com/packages/3d/vehicles/land/modern-sports-car-2196>
15. <https://assetstore.unity.com/packages/3d/environments/urban/lowpoly-modern-city-decorations-set-66070>
16. <https://assetstore.unity.com/packages/3d/environments/landscapes/lowpoly-environment-pack-99479>
17. <https://assetstore.unity.com/packages/3d/vegetation/lowpoly-vegetation-season-pack-lite-96083>
18. <https://assetstore.unity.com/packages/3d/environments/urban/lowpoly-modern-city-buildings-set-64427>
19. <https://docs.unity3d.com/ScriptReference/Input-gyro.html>
20. <https://docs.unity3d.com/Manual/GIIntro.html>
21. <https://www.kongregate.com/games/LongAnimals/drift-runners-2>
22. <http://www.hutchgames.com/smash-cops-heat/>