Dokumentation für Mutiplayer-Autorennspiel “Drift”

Informatik Leistungskurs 2

Zai Shi

Geliederung

1. Einleitung

Das Spiel “Drift” ist ein Autorennspiel für mehrere Spieler in einem LAN. Das Spiel wurde mit der Unity-Engine [1] programmiert und basiert auf C#.Unity ist ein sehr einfach zu benutzende Spiel-Engine, man kann 3D und 2D Spiele erstellen und rendern. Es hat auch sehr viele nützliche Funktionen zur Verfügung, wie zum Beispiel Physik, Belichtung,Netzwerk und GUI. Es läuft auf Android und Windows. Die Modelle wie die Häuser und Autos sind aus dem Unity-Assetsstore [12].
Ich mag Vogel perspektivische Autorennspiel wie Drift-Runner2[20] oder Smash-Cops-Heat [21] sehr.

2. Physik

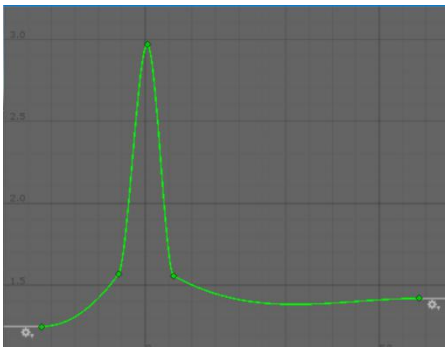
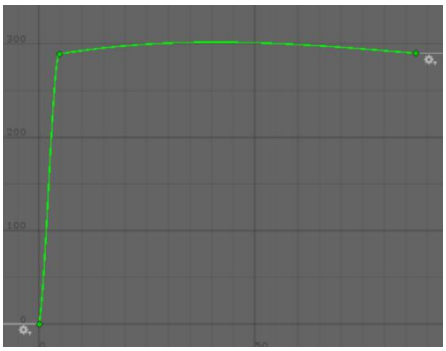
Die Berechnungen der Physik des Autos funktioniert mithilfe dreidimensionaler Vektoren(Vector3 [2]) im Script CarController. Die physikalischen Attribute des Autos sind:

Masse (float mass)
Bewegungsrichtung und Geschwindigkeit (Vector3 velocity)
Position (Vector3 position)
vorwärts (Vector3 forward)
aufwärts (Vector3 upward)

Ein Auto hat auch einige Bewegungsattribute. Eine AnimationCurve [3] ist eine Kurve, die man direkt in der Grafik bearbeiten kann. Folgendes sind Bewegungsattribute des Autos:

Beschleunigung/rückwärts Beschleunigung (float acceleration/backAcceleration)

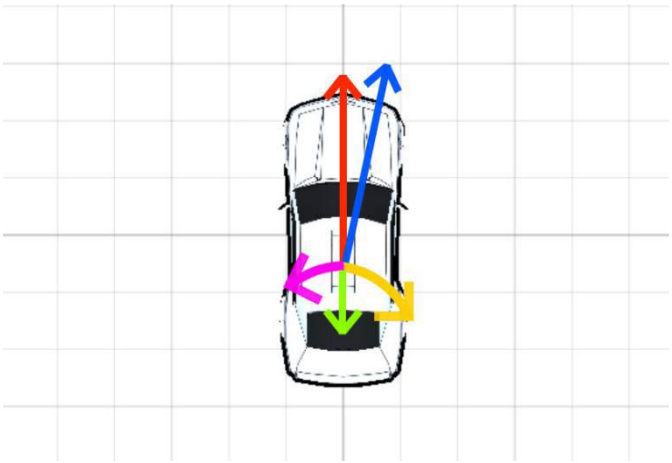
Trägheit der Beschleunigung (AnimationCurve accelerationDrag)
Beschleunigung der Drehung (AnimationCurve torque)
Trägheit der Rotation(float rotateDrage)

accelerationDrag und torque

Die Berechnung der Bewegung:

velocity += InputY() * forward * acceleration * deltaTime (deltaTime ist die Zeit zwischen jedem Frame, InputX()/InputY() ist der horizontale/vertikale Input des Spielers zwischen -1 und 1)
AddTorque (InputX() * upward * torque.Evaluate(velocity.magnitude) * deltaTime) (AddTorque() ist eine 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 Vektors)



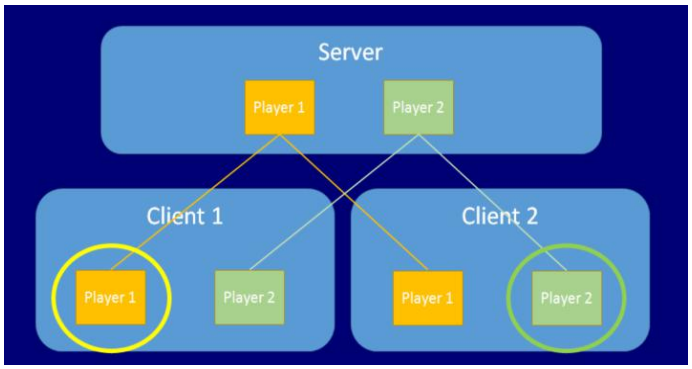
Rot: anfangs velocity, Grün: accelerationDrag, Gelb: torque, Lila: torqueDrag, Blau: neue velocity

Nach jedem Frame wird die Reibung berechnet:

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

3. Netzwerk und Syncronisierung

Ich habe die Netzwerk-Module UNet für mein Projekt benutzt. UNet ist ein high-level-scripting-API [5]. In UNet ist der Client der Spieler auch zugleich ein Server. Unet kontrolliert den Netzwerk mit einem NetworkManager [5]. UNet kann auch die Informationen und Befehle zwischen den Spieler versenden.



Server-Client Diagramm für UNet. Die Spieler in den Kreisen sind die Spieler, die von dem Client kontrolliert wird[6]

Wenn Spieler im LAN spielen kennen sie nicht die IP-Adresse der anderen Spieler, deshalb braucht man einen Script, der die IP-Adressen zu allen Ports des LANs überträgt, und die IP der anderen empfängt.

Im Hauptmenü (MainMenu) ist ein Script namens ConnectionManager, welcher NetworkDiscovery [7] erbt. Dieser hat vier Funktionen: StartServer(), StartClient(), CloseConnection() und überschreibt die Funktion OnReceivedBroadcast(). Die ersten drei Funktionen können einen Server oder Client für den IP-Broadcast erstellen und schließen, die letzte Funktion kann die IP-Adresse empfangen und behandeln.

In der Hauptszene (Town) habe ich zwei Scripts für das Netzwerk. Das erste ist der NetworkManager. Das zweite ist die AutoConnection, welche mit dem API von NetworkManager die IP-Adresse des Hauptmenüs empfängt und entweder mit dem Server verbindet, wenn der Spieler ein Client ist oder ein Server erstellt, wenn der Spieler ein Server ist.

Der CarController erbt NetworkBehavior [8]. Die kontrolliert nicht nur die Physik des Spielers, sondern auch die Synchronisierung der Positionen und Bewegungen. Es hat zwei Funktionen namens RpcSyncPos() mit den Attributen [ClientRpc][9] und CmdSendServer() mit den Attributen [Command][10], CmdSendServer(). Es schickt die Position, Rotation, Geschwindigkeit und ID der Spieler von einem Client zum Server, und auf dem Server wird RpcSyncPos() aufgerufen und die Informationen wird zu anderen Clients weiter geleitet. Wenn die anderen Clients die Informationen bekommen, dann wird LerpPosition() aufgerufen, die die Position der Spieler mit Lerp() [11] flüssig auf die neue Position stellt.

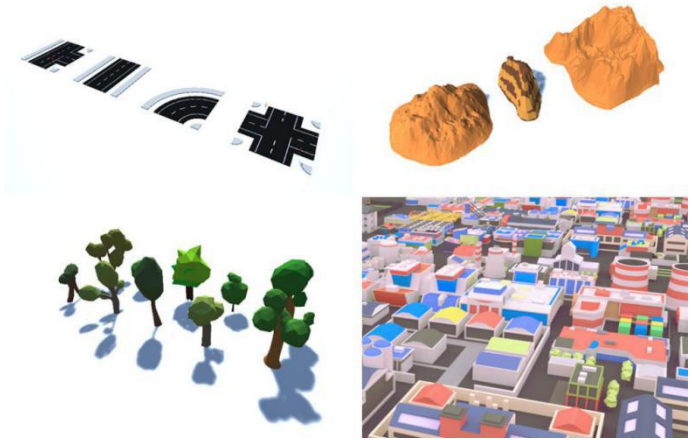
Zwei Spieler in einer Szene

4. Graphik und Modell

Die Modelle sind alle kostenlos vom AssetStore.
Die Automodelle [13] sind von “Modern Sports Car”, Ich habe die Texturen ein bisschen verändert, damit es besser zur Umgebung des Spieles passt. Außerdem habe ich eine Sirene auf das Polizeiauto gesetzt und die Bewegung der Reifen animiert.

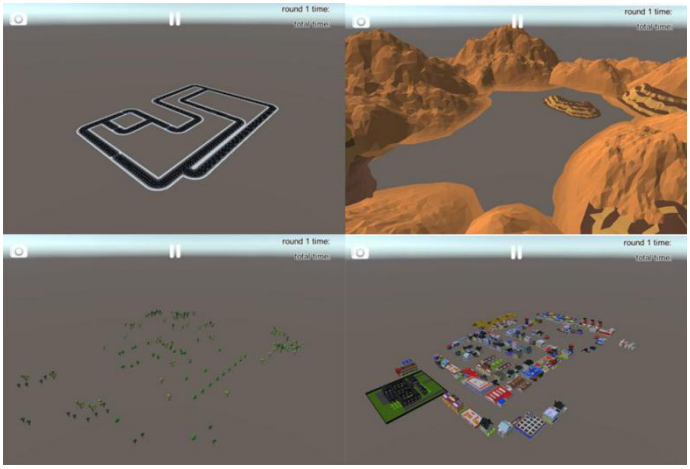
Automodelle

Dier Straßen-Asset ist von “Lowpoly Modern City Decorations Set” [14]. Die Bergmodelle kommen 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 mein Projekt in richtige Größe und Positionen eingestellt und eine kleine Stadt bebaut.

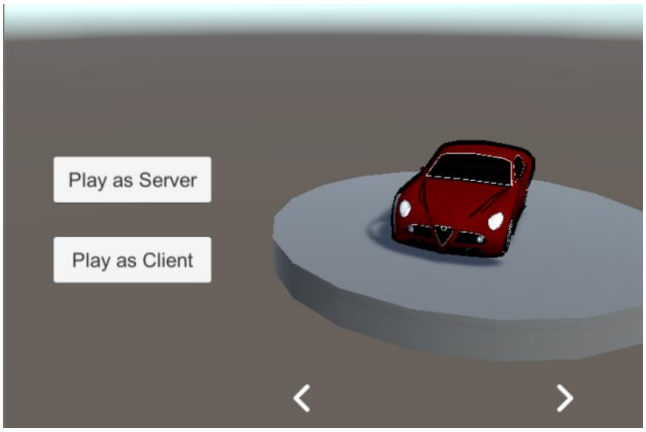


Straße, Berge, Bäume und Gebäude in der Szene

Überblick der Hauptszene (Town)

5. UI und Interaktion

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



Hauptmenü (MainMenu)

Wenn man auf Play klickt, gelangt man auf nächste die Seite, wo andere Spieler oder man selbst warten kann. Wenn der Client sich mit den Server verbindet, zeigt der Client auf dem Bildschirm die IP-Adresse des Servers an. Danach könne die Spieler auf Ready klicken, um das Spiel zu starten.

UI für Warten und Verbinden

Es gibt zwei Perspektiven im Spiel. Eine von oben und eine von hinten. Man kann diese mit dem Kamera-Knopf wechseln. Auf Windowsgeräten kontrolliert man das Spiel mit WASD-Taste oder ↑←↓→-Taste. Auf dem Android-Handy steuert man das Auto mit der Bewegung des Handys (Schwerkraft-Sensor Gyro() [18]). Die linke Seite des Bildschirm beschleunigt und die rechte Seite bremst und ermöglicht das Rückwärts fahren.

Unterschiedliche Perspektive

6. Fazit

Es gibt noch viele Sachen in dem Spiel, die man noch verbessern könnte. Das Ziel des Spieles wurde aufgrund zeitlicher Einschränkung nicht erreicht. Das Spiel sollte nach 3 oder 5 erreichten Runden eine Ziellinie zeigen, welche den schnellsten Spieler anzeigt. Die Grafik kann auch verbessert werden. Die Reflexion der Texturen, das Licht (Global Illumination [19]) und die Leistung könnten ebenfalls verbessert werden. Das Spiel erreicht zurzeit nur 60 FPS auf Core-M mit 1080x1920 Auflösung.

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

7. Quellen:

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