Godot slide

* Open-source game engine
* Programming language: GDscript (similar to Python)
  + Also supports C#
* Primarily used for graphics and UI
* Godot’s physics engine was not used

Efter att vi modellerat tyget i Matlab gick vi vidare till att skapa en animation i Godot, som är en open-source spelmotor. Arbetsflödet i Godot är ganska likt Unity, men det är ett generellt lättviktigare program som samtidigt har allt vi kände att vi behövde för projektet.

Godot har, som de flesta spelmotorer, en egen, inbyggd fysikmotor. Vi har inte använt den, all fysik är hanterad via originell kod som vi har skrivit. Detta var nödvändigt för att allt vi gjorde skulle stämma överens med vår förstudie i Matlab. Den huvudsakliga poängen med att använda Godot är att den gör det väldigt lätt att rita grafik i realtid, samt att göra gränssnitt som vi kan använda för att styra animationen.

( Demo )

Settings: 120/40/0.5

Här är vår implementation i Godot. Vi har alltså inte gjort en enda statisk animation utan ett gränssnitt där en användare kan ställa in parametrar och se hur tyget reagerar i realtid. Till att börja med har vi bara punkterna utritade, men vi kan också rita linjer mellan dom, eller rita ut en textur som täcker tyget.

Vi kan välja några punkter som statiska, dvs. Dessa påverkas inte av gravitation och kommer aldrig att flytta på sig. Du kan se det som att tyget är spikat fast i en vägg i dom punkterna och resten av tyget hänger löst. Om vi väljer andra alternativet här så kommer tyget hänga från två statiska punkter, upphängningspunkter om du vill. Nu tänjs tyget ganska mycket så vi kan öka fjäderkonstanten lite. I nedre högra hörnet har vi en widget som låter oss simulera vind. Vi kan rita ut en vektor i panelen, och vektorns storlek och riktning bestämmer vindens kraft och riktning.

Vi har två integraitonsmetoder, Euler och Verlet, som vi kan växla mellan fritt. Verlet ska, enligt den litteratur vi läst, vara mer stabil, ge ett mer realistiskt resultat och vara lika beräkningsmässigt krävande som Euler.

Verlet ser mer realistisk ut. Om vi gör tyget längre så ser vi att svängningar verkar propagera bättre genom tyget. Om vi byter tillbaka till Euler, så verkar nederdelen av tyget vara ganska stelt och inte svänga så mycket.

(120/40/0.5)

Vi kan se om vi kan illustrera skillnaden i stabilitet mellan Euler och Verlet. Både Euler och Verlet-integration har en steglängd, som bestämmer hur beräkningsmässigt intensiv fysikberäkningen blir och även hur stabil den är. Euler ska, enligt literatturen vi läst, kräva en mycket kortare steglängd än Verlet för att förbli stabil. Vår steglängd beror på vår “physics frames per second”, det vill säga, hur många gånger per sekund vi beräknar positioner för våra partiklar. Just nu är detta inställt på 60, dvs. Vi har en steglängd på ungefär 17 millisekunder.

Om vi reducerar physics FPS kommer steglängden att öka. Eventuellt når vi ett tröskelvärde där den numeriska integrationen blir instabil och tyget exploderar. Exakt var denna tröskel hamnar beror på vad vi har för inställningar på tyget, men vi kan se att med dessa inställningar så blir Verlet instabil runt 38 FPS, medan Euler blir instabil omkring 49 FPS.

4:15