Dokumentasjon: PG2101- Unity Hjemmeeksamen – vår 2015

Teoridel

1. C# og Java er begge er objektorienterte programmeringsspråk og har mange likheter, men det har blitt større forskjeller i de siste versjonene. Tidligere hadde Java mer funksjonalitet enn C#, men dette har i det siste blitt motsatt. En av de største forskjellene er at Java kan kjøres på andre platformer/operativsystem, mens c# kan kun kjøres i Windows, så det er viktig å tenke på hvilke platformer man skal lage programmet for før man velger en av disse.
   Noen ting C# har som ikke Java har:
   - God integrering med windows
   - Med Lamdas og LINQ støtter den litt funksjonell programmering.
   - typen dynamic som er «duck-typing» (å kjøre metoder på et objekt uavhengig av hvilken type objektet ble opprettet som.
   - Bedre enumeration støtte med yield statement.
   - Man kan definere nye verdi typer (non-reference)

   Noen fordeler Java har:

   - Bedre enums. Java behandler enums som objekter med oppførsel, istedenfor bare å gi deg en int som i C#.
   - Bedre IDE
   - Kan brukes på flere platformer, som mac OSX og Linux.

2. En Transform brukes for å lagre eller endre posisjon, rotasjon og skala/størrelse til et objekt. Alle objekter i en scene har en Transform, og et objekt i en scene er et GameObject. Transforms kan ha en parent, slik at posisjoner, rotasjon og størrelser til childs kan påvirkes i forhold til disse (dersom local space benyttes, se 3). For eksempel dersom en parent GameObject har posisjon -10 i x aksen vil en child GameObject under denne være i posisjon -10 i x aksen dersom dens posisjon er 0 i x aksen. Skriver man da 5 i x aksen på dette child objektet vil den være i posisjon -5 i spillet.

3. Global-rom og lokal-rom (Global space and Local space) er hvordan et objekt posisjoneres, roteres og endres i størrelse, i forhold til rommet/space som er valgt. I globalt rom er det i forhold til globale xyz aksene. I lokal rom er det i forhold til objektet, så da vil disse verdiene til ojektet påvirkes hvis noe lenger opp i Transform hierarkiet har andre verdier. Dersom en har valgt local og en parent har endret rotasjon så vil en child til denne få samme rotasjonen og rotasjonsverdier på denne vil være i forhold til parent sin.

4. Standard metoder som er med fra før når man oppretter et nytt script i monodevelop er Start og Update, men de tre vanligste metodene å bruke er nok Awake, Update og FixedUpdate.
   Metoden Awake kalles med en gang spillobjektet dette scriptet tilhører blir lagt til i scenen i løpet av eksekveringen av spillprogrammet. Den kjøres altså når scriptet «våkner», før andre metoder. I denne kan man finne andre objekter man trenger å gjøre ting med i scriptet, hente eller sette startverdier o.l.
   Metoden Update kjøres en gang per frame (bilde/skjermoppdatering). Hvor ofte denne kjører avhenger av hvilken FPS (frames per seconds / bilder per sekund) man har i spillet.

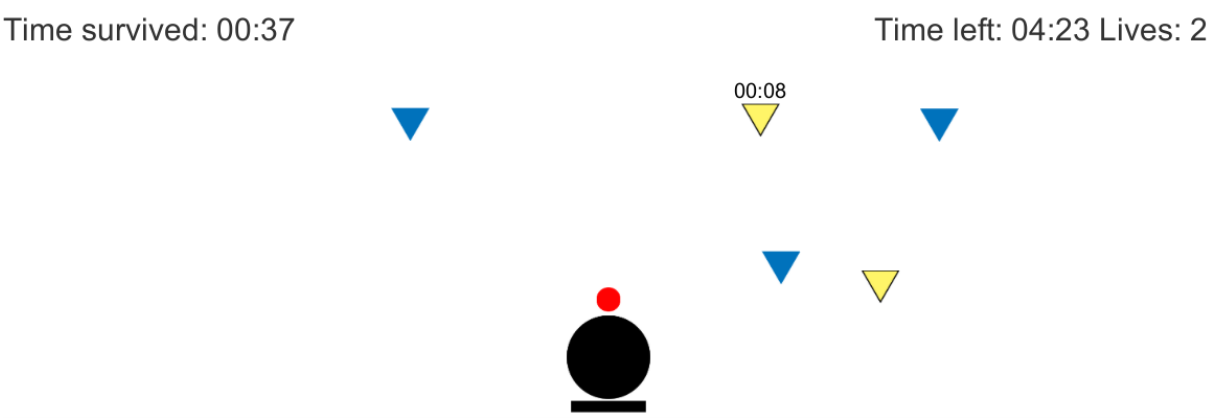
Denne bør brukes for ting man ønsker å sjekke eller endre i hver frame, for eksempel en posisjon.
Metoden FixedUpdate kjøres før hver fysikk oppdatering, og den kan kalles flere eller færre ganger per frame fordi den kjøres med en pålitelig tidtaker uavhengig av framerate. Fysikk oppdateringer skjer med en gang etter en FixedUpdate. Her trenger man ikke gange med Time.deltatime på det som har med bevegelses utregninger å gjøre, for eksempel hastigheten på en forflytning. FixedUpdate bør brukes for kode som har med fysikk å gjøre for eksempel ved bruk av rigidbody.

5. Koden kjører fordi det er ingen feil i koden, men koden har en mangel som gjør at den ikke får en effekt. Ny posisjon bare kalkuleres, objektet flyttes ikke. Først hentes nåværende posisjon i metoden Awake og lagres i Vector3 verdien _position. _position legger til 10f i x og y aksen fra Vector3 verdien _velocity og ganger denne med Time.deltaTime.
   Problemet er at scriptet endrer ikke posisjonen til objektet, det må den gjøre for å flytte objektet. Ny posisjon er kalkulert i linjen «_position += _velocity * Time.deltaTime;» i hver frame. For å sette den nye posisjonen må man legge til på neste linje i scriptet:
   transform.position = _position;
   objektet sin transform posisjon blir da endret og objektet flyttes 10f i x og y.

6. Time.deltaTime må brukes ved utregninger av bevegelser i Update metoden for å få en jevn hastighet. Verdien til Time.deltaTime er tiden det tok å fullføre forrige frame i antall sekunder. Update kjøres en gang per frame og hvis man ikke ganger med Time.deltaTime vil hastigheten variere ettersom hva framerate/fps i spillet er. Ved å bruke den blir spillet uavhengig av framerate og forflytningen blir 10f i x og y per sekund istedenfor per frame.

7. En prefab er en asset type for lagring et gameobjekt med tilhørende komponenter og innstillinger. Hvis man bruker en prefab til å lage mange av den same typen gameobjekt kan man gjøre endringer på prefaben og endringene vil skje på alle gameobjekts som har den tilknyttede prefab. Alle objekter som opprettes og gjenbrukes i en scene eller som det er flere av, burde ha en prefab tilknyttet. Alle objekter som er av samme type bør være knyttet til samme prefab så den har de samme innstillingene. Slik slipper man å gjøre samme endringen på en haug av objekter, og kan isteden bare gjøre endringen ett sted så oppdaterer den seg på alle. Med prefabs kan man lage nye instanser av et objekt i en scene. Det er mulig å overskrive komponenter og innstillinger for hver instans av en prefab uten at det påvirker de andre.

Praktisk del



Et skjermbilde av spillet

Balanseringsvalg

- Spilleren starter med 5 minutter, men tiden spilleren har øker ettersom hvor mange kurver man skyter og hvor mange ekstra sekunder man får på disse. Jeg synes 5 minutter var passe i forhold til vanskelighetsgraden for en nybegynner. Dette er jo første «level» og videre levler kunne da hatt kortere tid og raskere instansiering av objekter o.l. Spillet starter med 3 liv. Da kan spilleren bli truffet 3 ganger før spillet blir slutt. Det er vanlig med 3 liv i spill og jeg mener det passer bra i dette spillet også, i hvert fall med denne vanskelighetsgraden. I senere nivåer i spillet kunne jeg hatt færre liv for å gjøre det enda vanskeligere.
- Spilleren får 10 ekstra sekunder per blå kurv, fordi dette er de man skal skyte ballen oppi og den bør gi mer poeng. De gule gir en tilfeldig antall sekunder fra 1 til 5. Disse gir jeg mindre fordi du ikke må skyte ballen i dem og du mister ikke liv når de går ut av skjermen. Grunnen til at jeg tar tilfeldig tall fra 1 til 5 er for å gjøre det litt mer spennende.
- Hastigheten på laser-skuddet sender jeg med kraft 10f med 3f hastighet og bruk av Vector2.Lerp for mykere bevegelse og tregere bevegelse på slutten av skuddet.
- Spilleren flyttes halvparten så mye som kraft på skuddet, altså 5f, fordi laseren er gansk kraftig og spilleren skal kunne flytte seg litt raskt. Lerp brukes også her for jevnere bevegelse og saktere bevegelse nær høre og venstre side.
- Minimum hastighet for de røde ballene ved avfyring er 300f og maksimalt er 600f. Maks grensen er for at det ikke skal gå å skyte langt ut av spillet, og minstegrensen gjør det mulig å skyte en kort avstand ved å bare trykke uten krav om å holde space knappen lenge.
- De gule trianglene faller ned 10 sekunder etter at de dukker opp og det kommer en ny hvert 5. sekund. Dette gjør at spilleren må følge bedre med så han ikke mister liv og det gjør spillet mer utfordrende.
- Størrelsene på elementene har jeg beholdt samme størrelsesforhold som i spriten. Disse størrelsene virket naturlig og jeg synes de passet bra så det var ingen grunn til å endre disse.

Kort beskrivelse scripter

- GameManager kontrollerer spillinformasjon som hvor lang tid det er igjen, hvor lang tid spiller har overlevd, hvor mange liv som er igjen, skuddkraft, og viser disse til brukeren som tekst, og den starter gule trekant objekter.
- PlayerController styrer kontroll av spilleren med bevegelser ut fra tastetrykk.
- BallShoot bestemmer hastighet og skyter ballen mot riktig posisjon med hastigheten
- BasketBlue venter på at ballen skal treffe oppi den blå kurven med en trigger collider, så flytter den kurven nedover. Den ødelegger objektet om den blir skutt av laseren eller kommer helt ned.
- BasketYellow styrer de gule trekantene. Første teller den ned tiden til den blir 0 og så faller den nedover. Objektet ødelegges om den blir skutt av laseren eller den kommer utenfor skjerm.
- LaserShoot styrer skyting av laseren mot riktig retning og regner ut posisjonen den skal beveges til og flytter den mot dette punktet med Lerp for en mykere bevegelse.
- MenuControl henter highscore fra playerprefs og viser denne i menyen, og starter spillet ved trykk på play game knappen.
- PlayerCollision ødelegger objektene som kommer i kontakt med player ballen og sørger for at posisjonen til player ballen ikke flyttes.