

**Kode**

Spillet består av 8 scripts, 4 scener og diverse art-assets.

**ball.cs**Vi har brukt Unity sin innebygde fysikkmotor for å bruke RigidBody2D på ballen. Ballen har en utgangshastighet (ballInitSpeed) på 600f som skyter den avgårde, etter det vil ballen senke farten og lar fysikkmotoren bestemme hastighet. To if-statements vil også sjekke om ballen har en hastighet som er større/mindre enn ballMaxSpeed og ballMinSpeed og gjøre justeringer.

Om ballen har en y-posisjon som er mindre enn 4.25f, kalles funksjonen decreaseLife() i gameController hvor en enkel life variabel blir de-inkrementert med 1.

**brickSpawn.cs**Spillet har en scriptfil som automatisk plasserer brikkene på brettet. Vi vurderte dette som enklere enn å plassere dem for hånd. Senere la vi også til statiske powerups for å gi spilleren muligheten til å ødelegge brikker som gir ekstra liv, og som øker størrelsen på paddle

Vi laget en prefab av sprites som gjør at brikkene får 6 forskjellige farger. Scriptet spawner disse i seks rader hvor hver rad har sin egen farge. I scriptet blir et tilfeldig heltall mellom 0 (inklusiv) og 50 generert. En vanlig brikke spawnes om det tilfeldige tallet er mellom 1 (inklusiv) og 49. Er tallet 0, blir et nytt tilfeldig tall mellom 1 og 3 generert som bestemmer om spesialbrikken skal være «healthBrick» eller «paddlepower» Sjansen mellom disse er 50%, og sjansen for at en spesialbrikke spawnes er 1/50.

**brickControl.cs**Dette scriptet kaller på funksjoner fra gameController.cs og brickSpawn.cs  
Heltallsverdien for mode blir hentet fra tallgeneratoren i brickSpawn.cs, hvor verdien 1 som refererer til «healthBrick» kaller på funksjonen increaseHealth() i gameController.cs. Verdien 2 kaller på increaseScale() i paddleControl.cs

Hver brikke har en heltallsverdi health = 1, og når ballen treffer en brikke (spesial eller standard), blir verdien de-inkrementert til 0, som kaller på addscore() og destroyBrick() i gameController.cs

**gameController.cs**Dette scriptet holde rede på player score, funksjonen decreaseLife() som blir kalt i ball.cs, increaseLife() som blir kalt i brickControl.cs og en if-statement som laster inn scenen gameOver om life variablen er mindre eller lik 0.

**paddleControl.cs**Dette scriptet styrer paddle sin utgangsposisjon og hvor fort den beveger seg. Vi bruker klassen MathF. Clamp og transform innenfor Update() funksjonen for å styre paddle sin hastighet og akselerasjon.

Funksjonene decreaseScale() og increaseScale() blir kalt på henholdsvis i roofScript.cs og brickControl.cs.

**roofScript.cs**  
I dette scriptet blir funksjonen decreaseScale() i gameController.cs kallet på når ballen treffer taket.

**Kommentarer**

Det ble av forskjellige grunner vanskelig å avtale møtetidspunkter for jobbing, sånn at vi kommuniserte på Skype og sosiale medier. Vi brukte Github som en lagringsplass for prosjektfilene og er tilgjengelig [her](https://github.com/JesusFrelseren/Breakout-Clone---Gruppe-X-PG2201).

Scriptfilen for å spawne brikker var den vanskeligste delen, men også den delen som har det største potensialet. Vi har også hatt trøbbel med at UI ikke skaleres med oppløsning, men vi har gjort det slik at spillets optimale oppløsning er 1280x720.

Med mer tid og større kunnskaper, kunne vi ha utvidet dette brickSpawn.cs for å generere flere leveler og på sikt gjør det mer tilfeldig, men det å håndlage leveler i Breakout-kloner har også sine designmessige fordeler.

Vi hadde også latt oss inspirere av Arkanoid hvor powerups detter ut av brikker og må fanges opp av paddle, noe som gjør at gameplay blir mer utforutsigbart og fartsfylt.