Aflevering 3 - Asteroid flight

Information

Følgende aflevering skal laves individuelt og afleveres på lectio søndag d. 12/02 kl. 22.00

Produkt

I denne aflevering skal i lave et spil, hvor i styrer et rumskib der skal undgå asteroider.

Find template for koden i skal bruge i afleveringen her https://editor.p5js.org/fgcec/sketches/GD5YoBwas

Når i er på filen så tryk *File>Duplicate* for at kopiere skabelonen så den kan redigeres.

Format

Afleveringen kan afleveres som .pdf hvori der er et link til jeres kode, ELLER som en mappe hvori jeres kodefiler er indeholdt.

Kode i afleveringen

Angiv et link til jeres p5.js fil hvor opgaven er løst.

Opgave 3.1 - Indlæsning af billeder

I denne opgave skal du skrive kode til at indlæse de nødvendige billeder til projektet.
I mappen med denne opgave beskrivelse finder du følgende 4 billeder *coin.png, meteor.png, space.png* og *spaceship.png*.

Del 1 - indsæt billeder

Lav en mappe `resources` og upload til den, de vedlagte billeder.

Del 2 - `images`

Lav en ny global variabel `images` og initialiser den til et tomt objekt.

Del 3 - `preload`

Lav funktionen `preload`
https://p5js.org/reference/#/p5/preload

Beskrivelse
Funktionen skal ved brug af metoden `loadImage` indlæse de 4 billeder vedlagt opgaven ind i objektet `images` i et felt tilsvarende deres navn. Eksempelvis skal codeudsnittet `images.coin` returnere billeder af *coin.png*.

Del 4 - background

Set nu billedet *space.png* til at være baggrunden for hele kanvasset.

Opgave 3.2 - GameObject

Del 1 - Konstruktør

Beskrivelse

Lav en ny klasse `GameObject` i en separat fil med samme navn.
Nedenfor er følgende felter objektet skal have. Disse gives igennem konstruktøren med mindre andet et skrevet.

Felter

`x`: objektets x-værdi i planen

`y`: objektets y-værdi i planen

`r`: objektets radius

`xSpeed`: objektets hastighed langs x-aksen

`ySpeed`: objektets hastighed langs y-aksen

`isActive`: hvorvidt objektet er aktivt. Sættes altid til *true*

`sprite`: objektets billede. Gives igennem konstruktøren, men har en default værdi på *null*

`color`: objektets farve, givet det ikke har et billede. Er altid *"white"*

Del 2 - `intersects`

Beskrivelse

Metoden udregner hvorvidt det givene objekt kolliderer med objektet intersects kaldes fra.
For eksempel vi har to cirkler, cirkel1 og cirkel2. Hvis jeg kaldes metoden således `cirkel1.intersects(cirkel2.x, cirkel2.y, cirkel2.r)`, så returneres `true` hvis cirklerne kolliderer/overlapper og `false` hvis ikke det er tilfældet.

Parametre

`pointX`: x-koordinaten af objektet der tjekkes imod

`pointY`: y-koordinaten af objektet der tjekkes imod

`radius`: radiusen af objektet der tjekkes imod

Retur værdi

true or *false*

Del 3 - `move`

Beskrivelse

Funktionen *move* tilføjer objektets hastighed til dets placering for at bevæge det. Hvis objektets *y* er over *height* skal objektets status *isActive* sættes til *false*

Retur værdi

None

Del 4 - `display`

Beskrivelse

Denne funktione tegner objektet. Hvis *this.sprite* ikke er *null*, tegnes objektets *sprite* med *imageMode(CENTER)*, ellers tegnes objektet som en *circle* med dens *color*.

Retur værdi

None

Del 5 - `collision`

Beskrivelse

Når denne funktion kaldes, skal objektets *isActive* sættes til *false*

Retær værdi

None

Opgave 3.3 - Nedarvning

Del 1 - `Player`

Beskrivelse

Lav en ny klasse *Player* som arver fra *GameObject*

Konstruktøren skal tage argumenterne `x, y, r, sprite = null` som tilsvarer dem af samme navn i *GameObject* og kalde *super* med disse.

Ydermere skal *Player*s felt *color* initialiseres til *"green"*

Del 2 - `Player.move()`

Beskrivelse

Overskriv funktionen *move*, således den sætter `x` til at være lig musens *x-koordinat*

Retur værdi

None

Del 3 - `Asteroid`

Beskrivelse

Lav en klasse *Asteroid* der nedarver fra *GameObject*.
Den skal tage alle de samme argumenter i konstruktøren og kalde *super* med disse.

Ydermere skal den initialisere *color* til *"red"*

Del 4 - `Coin`

Beskrivelse

Lav en klasse *Coin* der nedarver fra *GameObject*
Den skal tage alle de samme argumenter i konstruktøren og kalde *super* med disse.
Ydermere skal den initialisere *color* til *"gold"*

Del 5 - `Coin.collision()`

Beskrivelse

Overskriv collision funktionen for *Coin*, således at der ved kollision tjekkes om objektet er aktivt, og hvis ja sættes *isActive* til *false* same der ligges 5, til variablen *score*. (*score* er ikke introduceret endnu, den kommer senere.)

Opgave 3.4 - Diverse

Del 1 - `spliceAll`

Beskrivelse

Lav en funktion `` `spliceAll `` der fjerner alle elementerne fra et array givet ud fra et andet array.

Parametre

`array`: At array af elementer hvorfra nogle skal fjernes

`splices`: Et array af indexes, som indeholder index til alle de elementer der skal fjernes fra *array*

Retur værdi

None

Del 2 - `tickScore`

Lav en ny global variabel *scoreTick* og initialiser den til 2500.

Beskrivelse

Hvis *millis()* overskrider *scoreTick*, skal der lægges 1 til variable *score* og 2500 til variablen *scoreTick*

Del 3 - `tickDifficulty`

Lav en ny global variabel *difficultyTick* og initialiser den til 5000.

Beskrivelse

Hvis *millis()* overskrider *difficultyTick*, skal der lægges 1 til variablen *difficulty* og 5000 til variablen *difficultyTick*

Del 4 - `drawUI`

Beskrivelse

Denne funktion skal skrive en tekst i øverste venstre hjørne, som viser hvad den nuværende *score* er.
Ydermere hvis spillet er slut, skal der skrives "Game Over" i midten af skærmen.

Kald denne funktion i slutningen af *draw*

Opgave 3.5 - Spil-logik

Del 1 - `setup`

Beskrivelse

Lav en global variabel *player* og initialiser den i funktionen `setup`, til et nyt `Player` object med en `x` = halvdelen af kanvas bredden, `y`=højden af kanvas - 50, samt en radius på 20 og giv den et billede af *spaceship*.

Del 2 - `is game over?`

Beskrivelse

Tilføj en global variabel `gameOver` som er initaliseret til *false*

Inde i funktionen `draw`, tjek nu om spillet er færdigt og hvis ikke, så bevæg og tegn playeren

Del 3 - `spawnNewObject`

Beskrivelse

Denne funktion laver et nyt object af *type* og tilføjer det til *objects* array som eksisterer globalt. Dette object skal laves med en tilfældig *r, x* og *ySpeed*, og dens *xSpeed* og *y* position starter som 0.

Såfremt der skal laves en "asteroid", bruges billedet *Asteroid.png* og hvis der skal laves en "coin" bruges billeder *coin.png*

Parametre

`type`: en string som beskriver hvilken type objekt der skal laves. Kan enten være "Asteroid" eller "Coin"

Retur værdi

None

Del 4 - `tickSpawning`

Beskrivelse

Denne funktion skal tjekke om tiden for programmer har oversteget henholdsvis *coinSpawnTick* og *asteroidSpawnTick*. Hvis *coinSpawnTick* er overskredet, skal der ved brug af metoden *spawnNewObject* laves et nyt *Coin* object, og *coinSpawnTick* skal stige med en random værdi fra 2000 til 4000.

Hvis *asteroidSpawnTick* er overskrevet, skal der ved brug af *spawnNewObject* laves antal nye asteroider svarende til *difficulty*. Ydermere skal *asteroidSpawnTick* derefter forlænges med en random værdi fra 100 til 500.

Del 5 - `tickObjects`

Beskriveslse

Denne funktion skal gennemløbe alle objekter i *objects*, derefter kaldes *move()* og *display()* på disse, samt tjekke om de kolliderer med player, hvis de kolliderer kaldes deres metode *collision()*. I tilfældet af en kollision med en asteroide skal spillet slutte.

Efterfølgende skal et objekt tjekkes om det er aktivt med *isActive*, hvis ikke skal det tilføjes til et array og ved brug af *spliceAll* fjernes fra listen af *objects.*