Exercise 4

Software Development 2018
Department of Computer Science
University of Copenhagen

Torben Olai Milhøj <vrw704@alumni.ku.dk>,
Emil Møller Hansen <ckb257@alumni.ku.dk>,
Casper Bresdahl <whs715@alumni.ku.dk>

GitHub:
https://github.com/Snapzie/su18-Casper-Emil-Torben

Version 1;
Due: Fredag, 2. marts

Contents

1 Opgave 4.1

1.1 Constructors til DIKUArcade.Windows

Classen Window har to forskellige constructors - i den første bliver Window instantieret med en titel (af typen string), en bredde "width" (af typen uint - ligger i intervallet 0 til $2^{32} - 1$) og en højde "height", ligeså af typen uint. Titlen fungerer som et navn på dette game window. Vinduet er et grafisk overlay, som er givet ved en højde og bredde, hvilket nemlig er denne bredde og højde, som vinduet initialiseres med. Netop derfor skal vinduets brede og højde intialiseres som typen "uint", da det ligger i intervallet 0 til $2^{32} - 1$ hvilket er passende for en egentlig længde, der naturligvis ej kan være mindre end nul. Desforuden sættes en boolean isRunning til true, idet et window initialiseres. Den anden constructor for window fungerer i høj grad på samme måde, idet den tager en titel som en string og en højde som en uint - men dets tredje argument, det initialiseres med, er af typen "AspectRatio", der er en enumerator med nogen forudbestemte størrelsesforhold - R1X1, R4X3 og R16X9. Initialiseres der et vindue ved denne constructor, sættes bredden enten lig højden (ved R1X1), hvilket medfører et kvadratisk vindue, bredden sættes lig $\frac{4}{3}$ gange højden eller $\frac{16}{9}$ gange højden. Disse forhold er meget almindelige, og vinduet kan derfor initialiseres således.

1.2 Constructors til DIKUArcade.Graphics.ImageStride

Den første constructor tager tre argumenter, idet objektet initialiseres - den tager argumentet "milliseconds" af typen int, som skal være større end 0 for ikke at kaste en undtagelse (en uint kunne være benyttet for at undgå, at denne undtagelse blev kastet under initialiseringen af objektet). Milliseconds svarer her til den frekvens, hvormed animationen for de forskellige grafiske objekter i spillet opdateres. Denne constructor tager ligeså som argument et string-array. Hvis længden af dette string-array er 0, hvilket tjekkes ved at sætte en int "imgs" lig længden af dette array, kastes en undtagelse, da der så ikke er nogle filer. I andre tilfælde laves en ny liste "textures" af typen "Textures", til hvilken der dernæst tilføjes alle Textures, som er at finde i den liste "imageFiles", som objektet initialiseres med. Dermed laves en liste af tilgængelige textures, som i spillet kan anvendes. Man kan ligeså benytte en anden constructor, til at initialisere et objekt af typen ImageStride, som tager og anvender en int "milliseconds" på samme måde, men tager som andet argument et "Image"-array, længden af hvilket atter skal være større end nul. Disse Images i Image-arrayet tilføjes nu en efter en til en liste "texture", atter af typen texture, hvilken Image initialiseres med, og som tilgås via funktionen GetTexture. Sidste constructor til ImageStride fungerer på præcis samme måde bortset fra, at

den her initialiseres med en liste af images, af typen Image, hvis texturer igen tilgås via funktionen GetTexture();

1.3 Inheritance, StationaryShape and DynamicShape

Både DynamicShape klassen og StaticShape klassen nyder godt af at nedarve fra klassen "Shape", fordelene af hvilket er mange - primært de mange skallerings- og bevægelses-muligheder, det tillader. Betragtes først klassen StationaryShape ses, at den besidder to constructorer - den første tager fire floats kaldet hhv. posX, posY, width og height. Denne constructor indeholder to fields, som har til formål at skabe position og extent som en vektor, hvor hver vektor skabes udfra et par af disse floats. Dette mellemtring kan springes over ved i stedet at instantiere klassen med to vektorer, i stedet for to par af to floats. Denne class indeholder desuden en method, der er en explicit operator, som har det formål at caste fra classen StationaryShape til classen DynamicShape. Det sker ved at returnere en DynamicShape, som tager den pågældende StationaryShape's variable, Position og Extent, der begge er vektorer, og instantiere en DynamicShape med disse. DynamicShape har flere constructors - den øverste fungerer meget ligesom den første constructor i StationaryShape, som tager fire floats hhv. posX, posY, width og height, og lave vektorer ud af disse. Hver constructor til DynamicShape har dog også endnu en method (Direction), der instantiers som en vektor uden bestemte koordinater. Constructor nummer to fungerer umiddelbart som den øverste, fordi den kalder denne constructor, men laver desuden to variable Direction.X og Direction.Y. Constructor nummer 3 fungerer præcis som den første bortset fra, at den tager to vektorer i stedet for 4 floats. Den sidste constructor tager tre vektorer således, at den tredje vektor instantieres som variablen "Direction". Hver constructor instantierer altså en variabel "Direction", der kan ændres som man lyster igennem funktionen "ChangeDirection". Denne variabel Direction benyttes i funktionen "Move", som overrider den oprindelige Move-funktion fundet i shape og tilføjer direction til den variabel Position, som tilhører superclassen Shape, af hvilken DynamicShape er en nedarvet klasse.

2 Vores implementation

2.1 Game speed og update rate

Vi har sat både updates per second (UPS) og frames per second (FPS) til 60. Vi bruger et GameTimer-objekt til at holde styr på tiden. Inde i et while-loop der kører så længe spillet gør kaldes MesureTime() og derefter tjekkes de tre boolske værdier, ShouldUpdate(), ShouldRender() og ShouldReset().
Så længe ShouldUpdate() returnerer sand vil vinduet se om der er nogle events, og derefter vil eventBus processere eventuelle envents. Når denne ikke læn-

gere er sand tjekkes ShouldRender(). Hvis denne er sand betyder det at det er tid til at gentegne vinduet og rykke objekter i vinduet. Derfor kaldes player.Move() for at rykke spilleren i den retning den er sat til. Derefter bliver hele vinduet tegnet sort for at fjerne gamle tegninger, hvorefter enemies, eventuelle eksplosioner og skud samt spilleren gentegnes i de nye positioner. Når disse er tegnet kaldes funktionen win.SwapBuffers() som er en del af OpenTK-biblioteket.
Til sidst tjekkes ShouldReset(), denne bliver sat til true når der er gået ét sekund, og når det sker vil vinduets titel opdateres til at vise den seneste frame- og update-rate.

2.2 Større designvalg for Player

Vores første tanker omkring at implementere Player var at vi skulle lave en Entity i constructoren. Dernest har vi en property som har en public get men en private set som er lig vores entity. På denne måde kan Game få vores players position, men kan ikke ændre på noget i Player objektet. Udover dette har vi fem metoder som alle var skrevet i Game men blev flyttet til Player. Ideen med dette er at Player nu er "ansvarlig" for sig selv, og Game blot fortæller Player objektet hvad den skal gøre, i stedet for at gøre det for den. I vores løsning for at holde Player objektet inde på skærmen tjekker vi under Move() om spillerens position er inden for skærmen. Hvis den ikke er dette så sættes dens position til at være det. På denne måde så vil spilleren hele tiden blive tegnet på det yderste punkt selvom den forsøger at bevæge sig længere ud.

2.3 Forbedringer

I vores arbejde med opgaverne har vi ikke implementeret noget på andre måder end som var beskrevet i opgaverne. Men i vores arbejde med opgave 3.7.1 endte vi med at checke for collisions mellem alle skud og alle enemies, og bagefter iterere over alle skudene igen for at finde ud af om de var out of bound. En nemmere løsning ville være at tjekke for dette inde i vores foreach loop, men for at slette skudene er det nødvendigt at implementere IteratorMethod delegate for at kunne kalde EntityContainer.Iterate. Derfor ville en bedre løsning have været at neste to delegate implementationer for at tjekke om et skud har en collision med en enemy og samtidigt tjekke for om skudet er out of bounds, og derefter slette alle objekter hvis isDeleted er true.