-Software Development Assignment 4-

Daniel Burgaard, Martin Henriksen, Dan Yu Wang, Lauritz Koch

8. marts 2019

Introduction

Denne rapport omhandler implementeringen af spillet Galaga. Implementationen er foretaget, som en abstraktion af det klassiske arkadespil Galaga, hvilket blev udviklet helt tilbage i 1981. Spillet er dog stadig populært. Opgaven som blev stillet inkluderede kritiske aspekter af Galaga, såsom at oprette en spiller, fjendtlige rumvæsener, samt at få spilleren til at flytte sig og kunne skyde efter fjender.

Spørgsmål om DIKUArcade

Formål for DIKUArcade.GameEventBus

The GameEventBus nedarver fra to interfaces, IGameEventBus<T> and IGameEventBusController<T>. Formålet med GameEventBus er at alle events lægges i en kø, hvorefter de håndteres samtidig istedet for at håndtere dem, mens de sker i GameLoop. Hvis der ikke var en GameEventBus skulle man altså selve håndtere sine events. Dette sker her automatisk i classen.

DynamicShape and StationaryShape

Dynamic shape og Stationary shape inheriter shape. Et entity objekt har en shape. For at tilgå Dynamic- eller Stationary shape downcastes entity objektets shape-field, som enten dynamicShape eller StationaryShape. I vores Galaga-implementation bruges downcastingen i form af 'Shape.AsDynamicShape()' for at tilgå 'player's direction, og derved flytte player til højre eller venstre.

Formål og forskel på DIKUArcade.Graphics.ImageStride and DIKUArcade.Graphics.Image

Klassen DIKUArcade.Graphics.Imagestrider er den dynamiske udgave af image, som kan agere som animation, hvor figuren selv opdateres løbende. I modsætning hertil er DIKUArcade.graphics.image statisk.

Spørgsmål om implementeringen af Galaga

Hvordan logic og rendering er konsistent på forskellige maskinger

Vores implementering sikrer, at det er den samme skalering og bevægelse som benyttes på tværs af maskiner uanset størrelsen på vinduet. Det skyldes at vi arbejder med et koordinatsystem fra 0-1 i begge retninger, og oversætter dette til

vinduets størrelse. Det er her ikke muligt at benytte et while-loop, da forskellige maskiner med forskellige hastigheder ville køre et forskelligt antal while-loops. Dette undgås ved at sætte et loft på Frames per second(FPS) og Updates per second(UPS).

Begrænsning af antallet af læsninger fra harddrivet

At 'opdatere' et billede for at lave fx en animation, kræver at man skifter mellem en serie af billeder. for at undgå at skulle allokere ny plads på harddriven for hver gang man skifter et billede, starter man med at allokere plads, og gemme billederne i en liste, for derefter at referere til listen når animationen skal udføres.

Brug af object-orienterede programmeringsprincipper i implementationen

Encapsulation benyttes ved både GameTimer, GameEventBus, Player og Window, som eksempelvis er gjort readonly. Inheritance benyttes både til at implementere player, enemy og shot, som nedarver fra Entity. Polumorphism bruges til at udbygge Shape-funktionen, når Shape-funktionen downcastes, så den modtager de samme egenskaber som DynamicShape eller StationaryShape.

Afvigelser i spillet fra hvad der var foreslået i opgaven

Der har ikke været betydelige afvigelser fra hvad der var foreslået i opgaven.

Conclusion

Vores implementation af spillet Galaga løser samtlige af de stillede spørgsmål i opgavebeskrivelsen. Igennem denne rapport besvares tekniske spørgsmål om brugen af den udleverede Gameengine DIKUArcade, samt implementationen af gruppens version af Galaga.