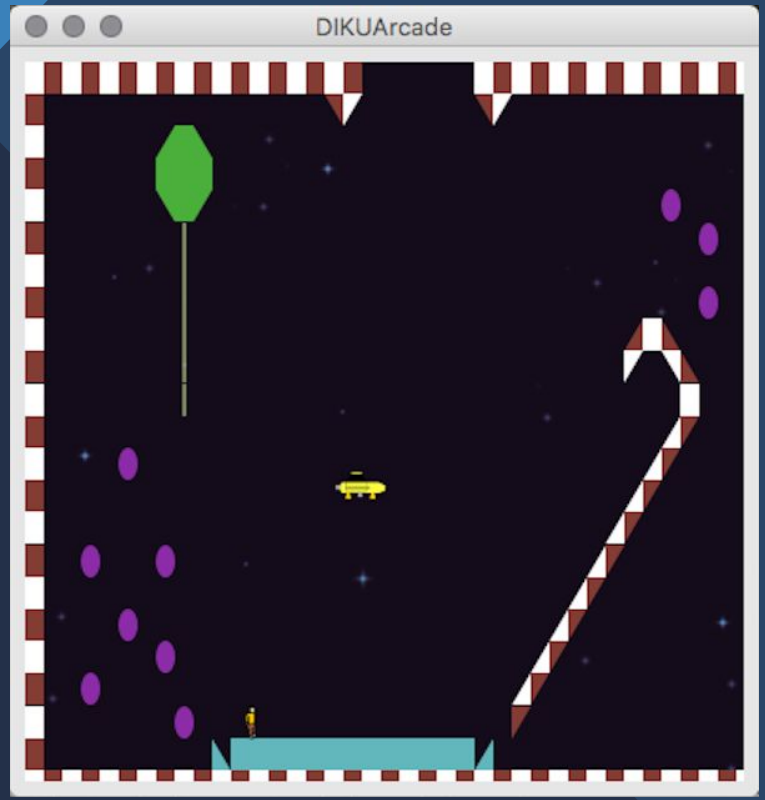
SPACE TAXI

Softwareudvikling

af Sarah Maria Hyatt, Københavns Universitet, juni 2017



INTRODUKTION

Arcade spil; Space Taxi

C#

Crossplatform

Deliverable 1, 2, 3 and 4

Indhold

Demo

Design

Implementation

Tests

Udviklings process

Fejl og forbedringer

Evaluering og konklusion

DEMO

DESIGN

SOLID-Principles

- Single responsibility
- Open-closed
- Liskov substitution
- Interface segregation
- Dependency inversion

Refactor cirkulær reference

- Entities og Game

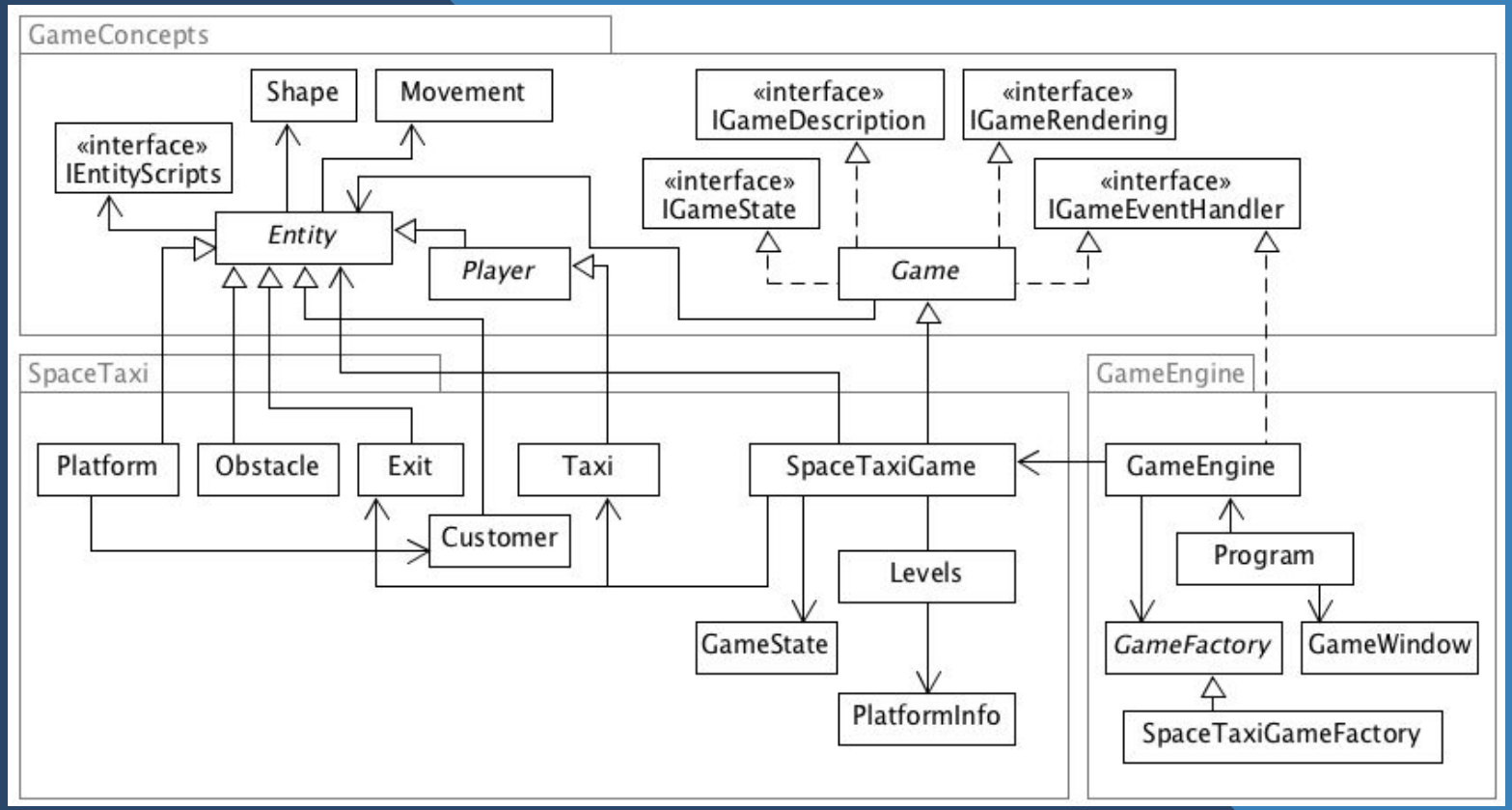
Singleton Pattern

- SpaceTaxiGame

Coupling

- Associations
- Inheritance
- Cirkulær reference

UML CLASS DIAGRAM



SINGLETON PATTERN

```
public static SpaceTaxiGame GetGame (Drawing.GameArea area)
{

   if (s_game == null) {
      s_game = new SpaceTaxiGame (area);
   } else
   if (!s_game._area.Equals (area)) {
      throw new InvalidOperationException ("SpaceTaxiGame not " +
                                           "initialized.");
   }
   return s_game;
}
```

IMPLEMENTATION (1/3)

```
public void CheckCustomerCollision (Taxi taxi)
{
    if (HasCustomer () == true) {
        if (taxi.Shape.OverlapsWith (_customer.Shape)) {
            _customer = null;
        }
    }
}
public bool HasCustomer ()
{
    bool has_cust;
    if (_customer == null) {
        has_cust = false;
    } else {
        has_cust = true;
    }
    return has_cust;
}
public void AddCustomer (Customer customer) { _customer = customer; }
```

PLATFORM & CUSTOMER

- Hænger sammen
- Random valg

IMPLEMENTATION (2/3)

COLLISION DETECTION

- Player vs. Objekter
- Exit; MoveToNextLevel

```
public override void PerformCollisionDetection ()
    {
        bool taxiLand = false;

        foreach (Entity e in _entities) {
            if (Player.Shape.OverlapsWith (e.Shape)) {
                if (e is Obstacle) {
                    RenderGameLost ();
                    _state = 2;
                    break;
                }
            }

        } else if (_exit != null) {
            if (Player.Shape.OverlapsWith (_exit.Shape)) {
                MoveToNextLevel ();
            }
        }
        HandlePendingEntities ();
    }
```


IMPLEMENTATION (3/3)

```
} else if (e is Platform) {
        if ((Player.Y < e.Y) &&
           _taxi.GetSpeed () <= Platform._maxSpeed) {
           taxiLand = true;
        } else {
           RenderGameLost ();
           _state = 2;
           break;
        }
        ((Platform)e).CheckCustomerCollision (_taxi);
     }
  }
}
if (taxiLand == true && landed == false) {
   _taxi.CounterMovement ();
   landed = true;
} else if (taxiLand == false && landed == true) {
   _taxi.AddGravity ();
   landed = false;
```

PLATFORM COLLISION

- Landings hastighed
- Ovenfor landing
- Customer Collision
-
- Kræfter fjernes
- Hastigheden sættes til 0
- AddGravitiy ved piletaster

TESTS

UNIT-TESTING

Game Tests

- Tester spillet
  - Entities og level
  - GameState

Level Tests

- Tester level parseren i forhold til ASCII-filerne
- MoveToNextLevel
- Korrekt level fra starten

Physics Tests

- Tester spillerens bevægelser:

  Op, ned, højre, venstre, op/højre, op/venstre og gravitation

DEVELOPMENT PROCESS

AGILE

Pairwise
Test-first

Arbejdet ud fra delmål i Deliverables

CONFIGURATION MANAGEMENT

Gitlab; DIKUnix

DOCUMENTATION

Doxygen

FEJL & FORBEDRINGER

COLLISION DETECTION

Polymorphism

DESIGN

Singleton

Composite fejl

UML havde fejl

PLATFORM

SolidSurface

Move

LEVELS

Cirkulær reference

Obstacles (Level parser)

TAXA

Taxa forces

Gravity

EVALUERING & KONKLUSION

Slutbrugeren

Crossplatform

Arbejde med DIKUArcade

Dokumentation

Vedligeholdelse og videreudvikling