SU

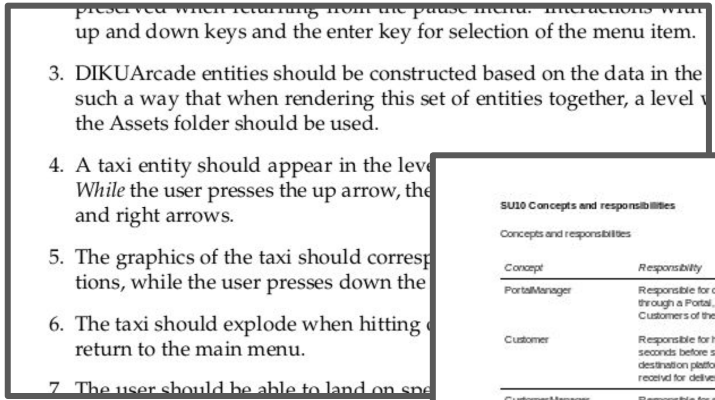
Kasper Rasmussen, wdq486

Lifecycle

Process

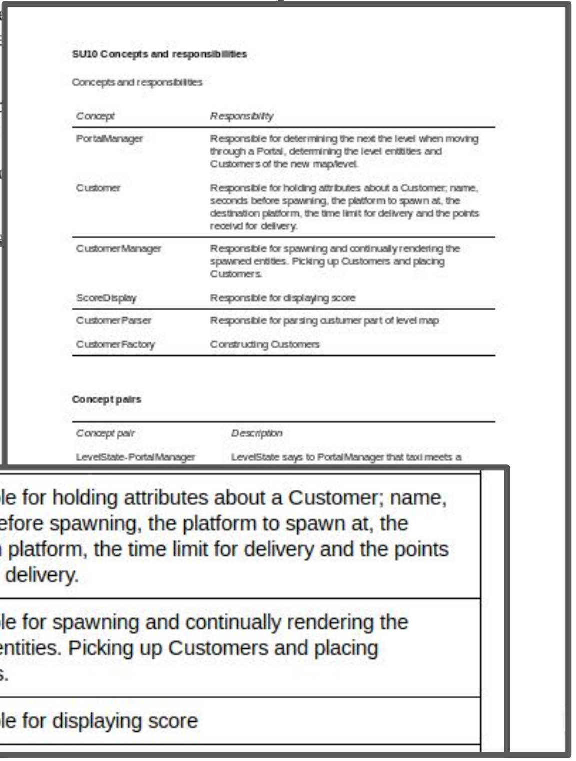
- 3 phases
- Lifecycle
- Analysis and design
- Modelling
- Low-level specification

Evaluation later



preserved when returning from the pause menu. Interactions with
up and down keys and the enter key for selection of the menu item.

3. DIKUArcade entities should be constructed based on the data in the
such a way that when rendering this set of entities together, a level
the Assets folder should be used.
4. A taxi entity should appear in the leve
While the user presses the up arrow, the
and right arrows.
5. The graphics of the taxi should corresp
tions, while the user presses down the
6. The taxi should explode when hitting
return to the main menu.
7. The user should be able to land on spe



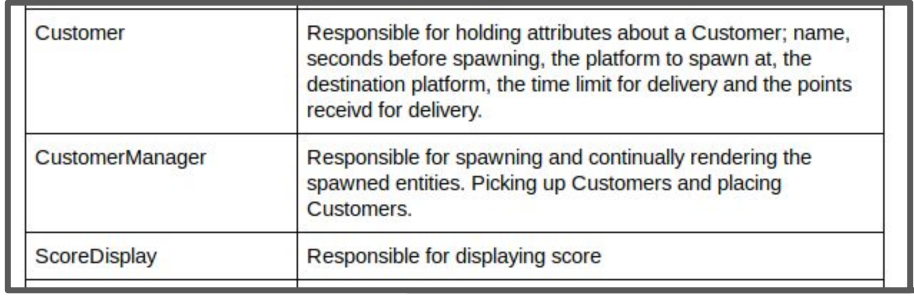
SU10 Concepts and responsibilities

Concepts and responsibilities

Concept	*Responsibility*
PortalManager	Responsible for determining the next the level when moving through a Portal, determining the level entitities and Customers of the new map/level.
Customer	Responsible for holding attributes about a Customer; name, seconds before spawning, the platform to spawn at, the destination platform, the time limit for delivery and the points receivd for delivery.
CustomerManager	Responsible for spawning and continually rendering the spawned entities. Picking up Customers and placing Customers.
ScoreDisplay	Responsible for displaying score
CustomerParser	Responsible for parsing custumer part of level map
CustomerFactory	Constructing Customers

Concept pairs

Concept pair	*Description*
LevelState-PortalManager	LevelState says to PortalManager that taxi meets a



Customer	Responsible for holding attributes about a Customer; name, seconds before spawning, the platform to spawn at, the destination platform, the time limit for delivery and the points receivd for delivery.
CustomerManager	Responsible for spawning and continually rendering the spawned entities. Picking up Customers and placing Customers.
ScoreDisplay	Responsible for displaying score

Design

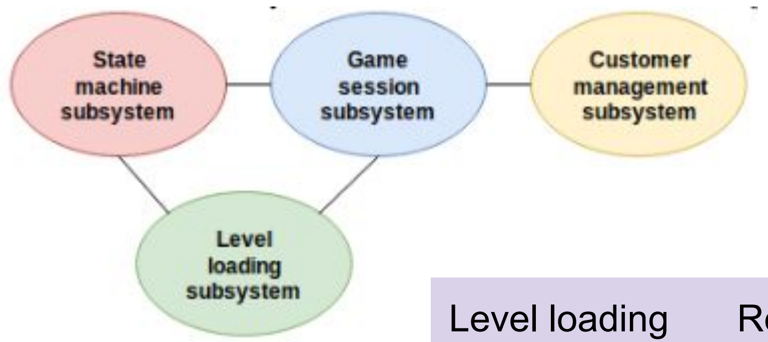
- Responsibility
- Modularity

SOLID

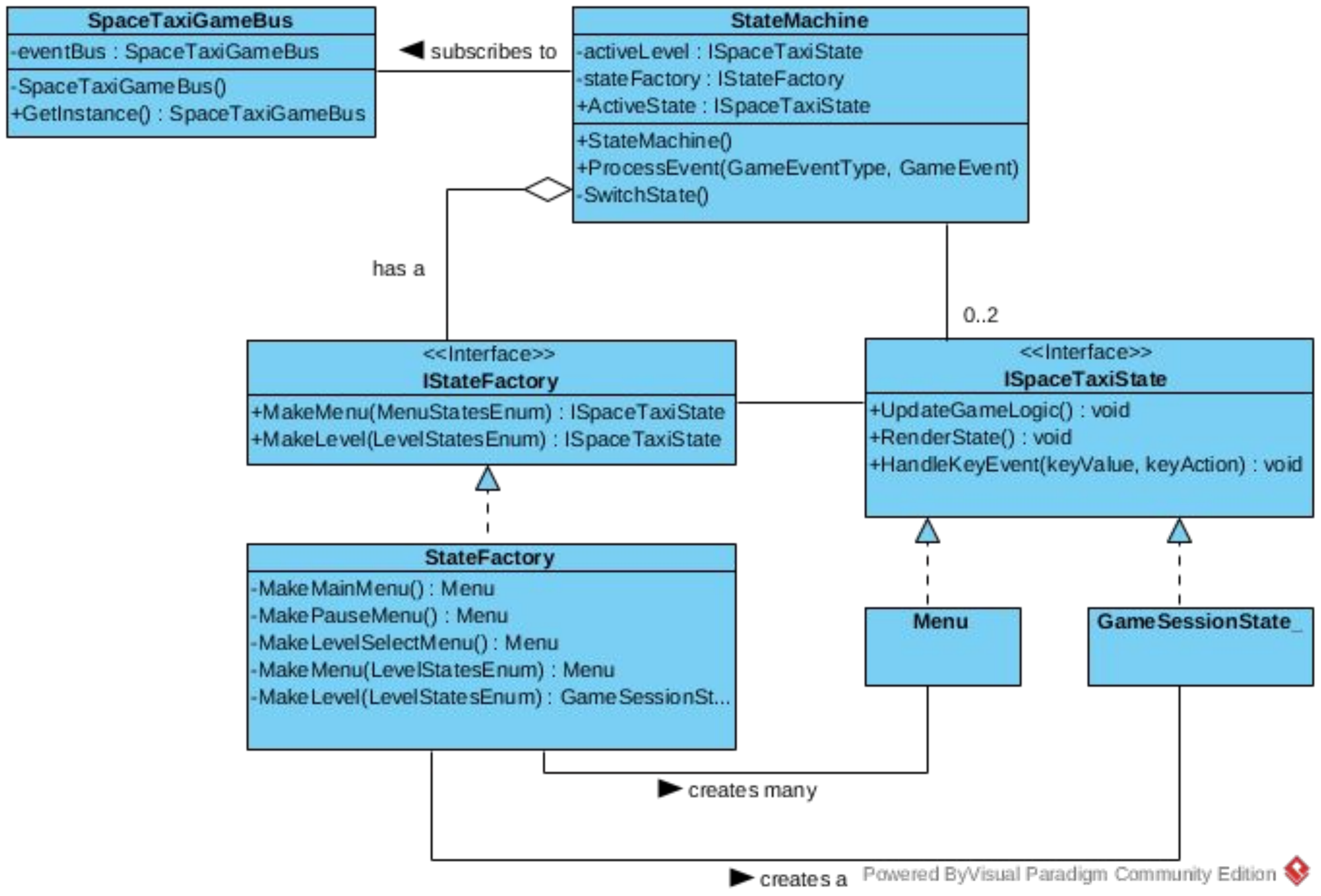
- Single responsibilities
- Open-closed
- Dependency inversion

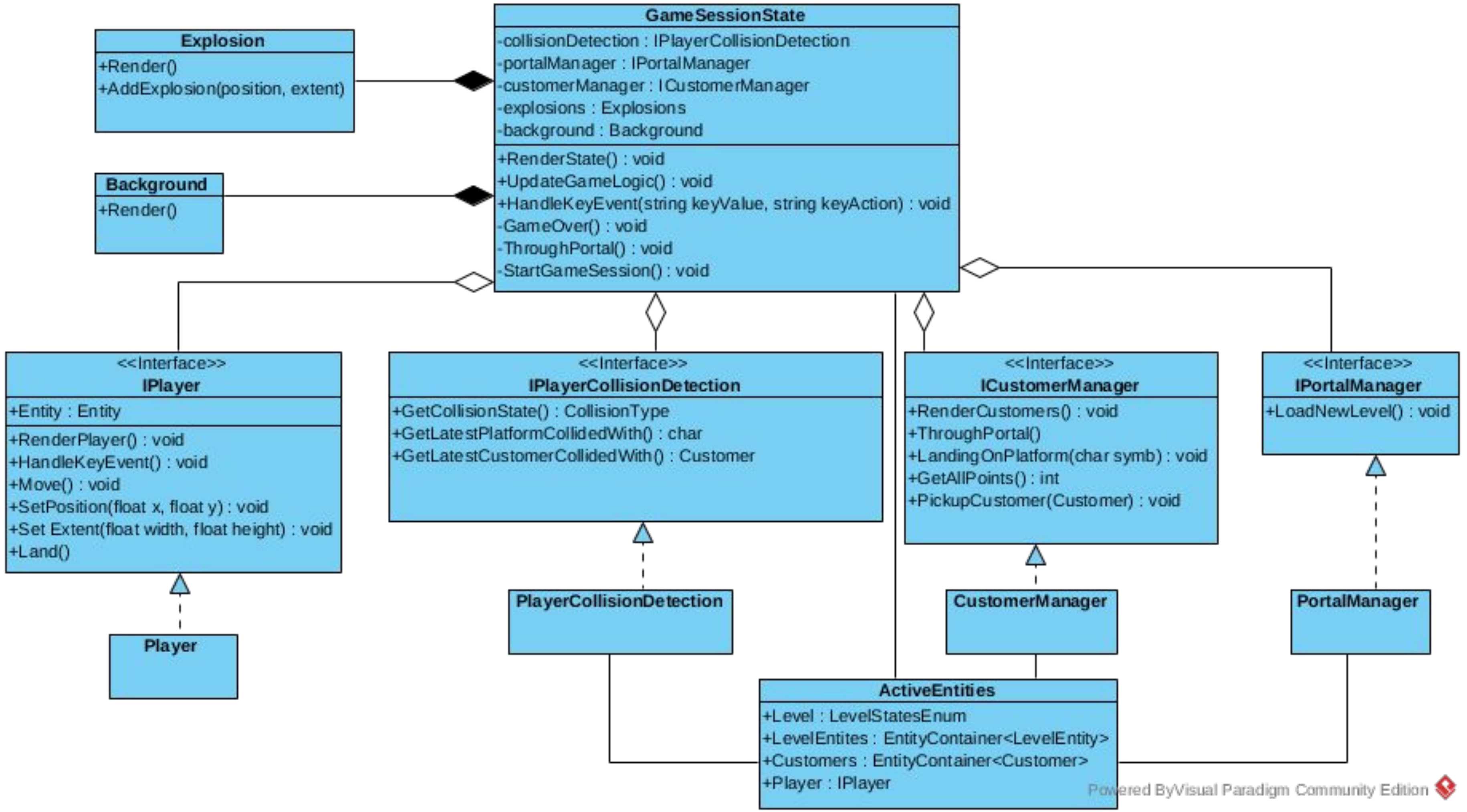
Patterns

- Singleton
- Mediator, observer
- Abstract factory



Level loading	Reqs. 3, 7-9
State machine	Reqs. 2
Game session	Reqs. 1, 4-8
Customer mng.	Reqs. 9-13


SpaceTaxiGameBus
-eventBus : SpaceTaxiGameBus
-SpaceTaxiGameBus()
+GetInstance() : SpaceTaxiGameBus
◀ subscribes to
StateMachine
-activeLevel : ISpaceTaxiState
-stateFactory : IStateFactory
+ActiveState : ISpaceTaxiState
+StateMachine()
+ProcessEvent(GameEventType, GameEvent)
-SwitchState()
has a
0..2
<<Interface>>
IStateFactory
+MakeMenu(MenuStatesEnum) : ISpaceTaxiState
+MakeLevel(LevelStatesEnum) : ISpaceTaxiState
<<Interface>>
ISpaceTaxiState
+UpdateGameLogic() : void
+RenderState() : void
+HandleKeyEvent(keyValue, keyAction) : void
StateFactory
-MakeMainMenu() : Menu
-MakePauseMenu() : Menu
-MakeLevelSelectMenu() : Menu
-MakeMenu(LevelStatesEnum) : Menu
-MakeLevel(LevelStatesEnum) : GameSessionSt...
Menu
GameSessionState_
▶ creates many
▶ creates a
Powered ByVisual Paradigm Community Edition



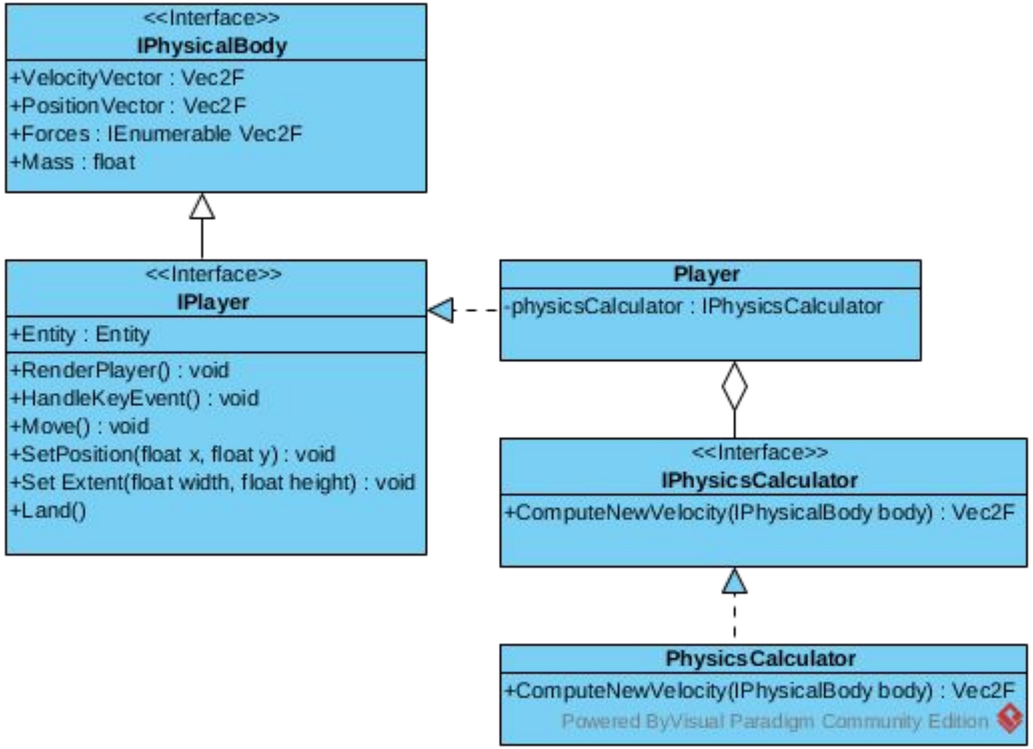
Explosion
+Render()
+AddExplosion(position, extent)
Background
+Render()
GameSessionState
-collisionDetection : IPlayerCollisionDetection
-portalManager : IPortalManager
-customerManager : ICustomerManager
-explosions : Explosions
-background : Background
+RenderState() : void
+UpdateGameLogic() : void
+HandleKeyEvent(string keyValue, string keyAction) : void
-GameOver() : void
-ThroughPortal() : void
-StartGameSession() : void
<<Interface>>
IPlayer
+Entity : Entity
+RenderPlayer() : void
+HandleKeyEvent() : void
+Move() : void
+SetPosition(float x, float y) : void
+Set Extent(float width, float height) : void
+Land()
<<Interface>>
IPlayerCollisionDetection
+GetCollisionState() : CollisionType
+GetLatestPlatformCollidedWith() : char
+GetLatestCustomerCollidedWith() : Customer
<<Interface>>
ICustomerManager
+RenderCustomers() : void
+ThroughPortal()
+LandingOnPlatform(char symb) : void
+GetAllPoints() : int
+PickupCustomer(Customer) : void
<<Interface>>
IPortalManager
+LoadNewLevel() : void
Player
PlayerCollisionDetection
CustomerManager
PortalManager
ActiveEntities
+Level : LevelStatesEnum
+LevelEntites : EntityContainer<LevelEntity>
+Customers : EntityContainer<Customer>
+Player : IPlayer

Implementation

- Concepts to classes
- Interfaces and properties
- Composition, constructor injection

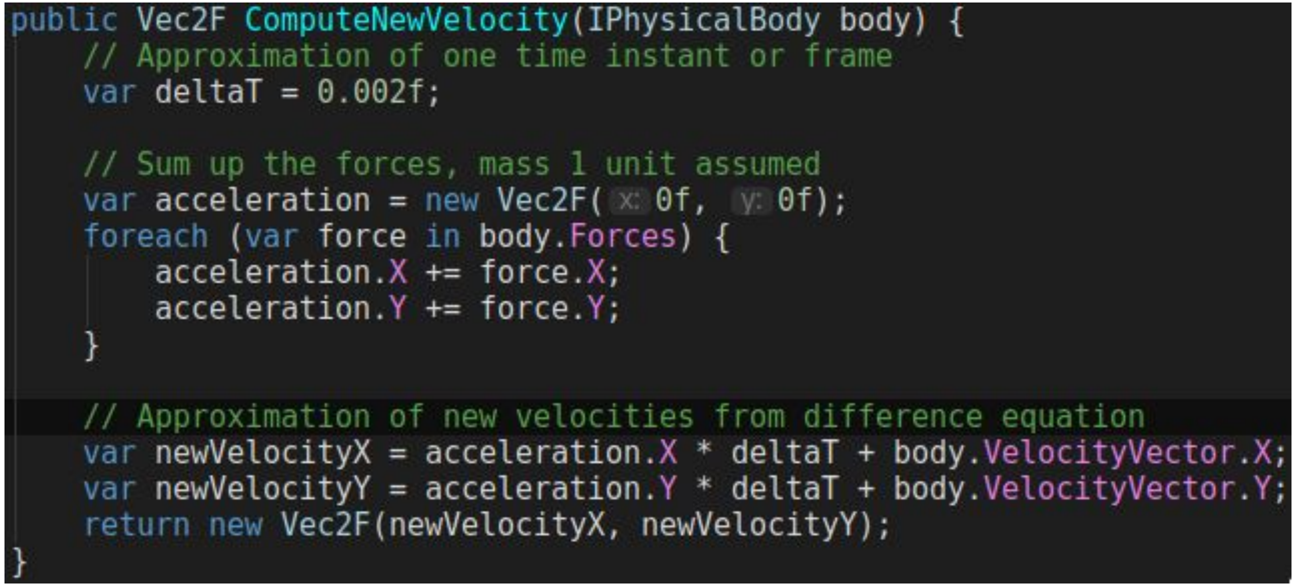
Physics (I)

REQ-4: A taxi entity should appear in the levels which is convincingly subject to Newtonian physics [...]



<<Interface>>
IPhysicalBody
+VelocityVector : Vec2F
+PositionVector : Vec2F
+Forces : IEnumerable Vec2F
+Mass : float
<<Interface>>
IPlayer
+Entity : Entity
+RenderPlayer() : void
+HandleKeyEvent() : void
+Move() : void
+SetPosition(float x, float y) : void
+Set Extent(float width, float height) : void
+Land()
Player
-physicsCalculator : IPhysicsCalculator
<<Interface>>
IPhysicsCalculator
+ComputeNewVelocity(IPhysicalBody body) : Vec2F
PhysicsCalculator
+ComputeNewVelocity(IPhysicalBody body) : Vec2F
Powered By Visual Paradigm Community Edition

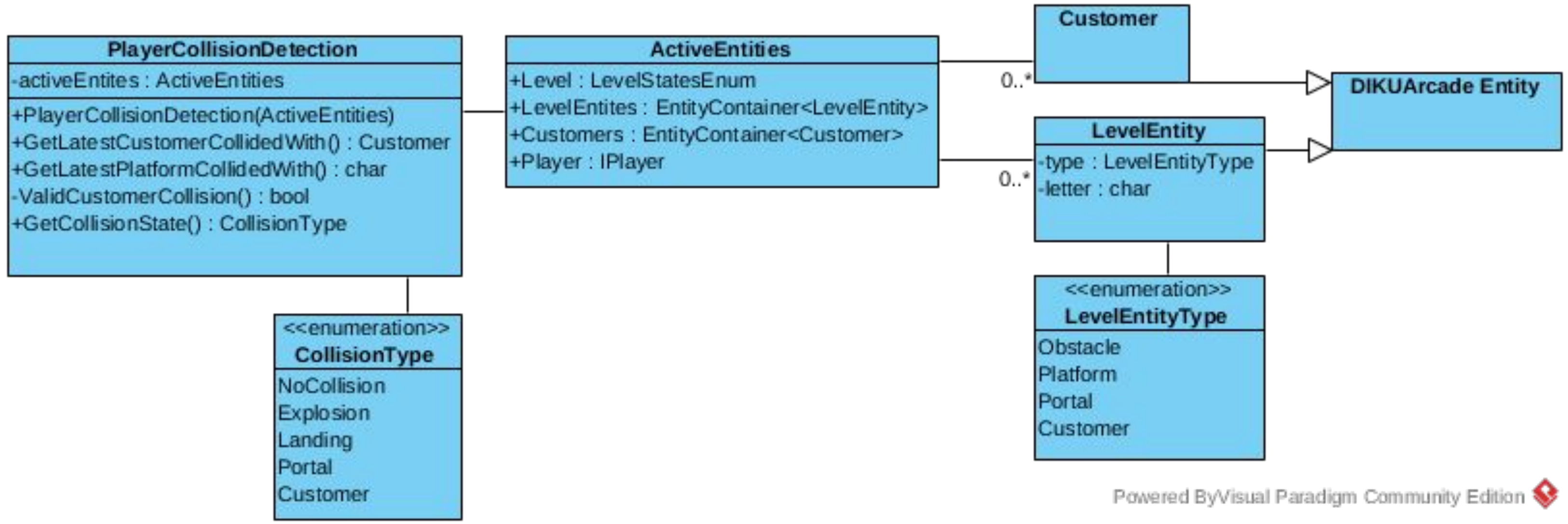
Physics (II)



```
public Vec2F ComputeNewVelocity(IPhysicalBody body) {
    // Approximation of one time instant or frame
    var deltaT = 0.002f;

    // Sum up the forces, mass 1 unit assumed
    var acceleration = new Vec2F( x: 0f,  y: 0f);
    foreach (var force in body.Forces) {
        acceleration.X += force.X;
        acceleration.Y += force.Y;
    }

    // Approximation of new velocities from difference equation
    var newVelocityX = acceleration.X * deltaT + body.VelocityVector.X;
    var newVelocityY = acceleration.Y * deltaT + body.VelocityVector.Y;
    return new Vec2F(newVelocityX, newVelocityY);
}
```

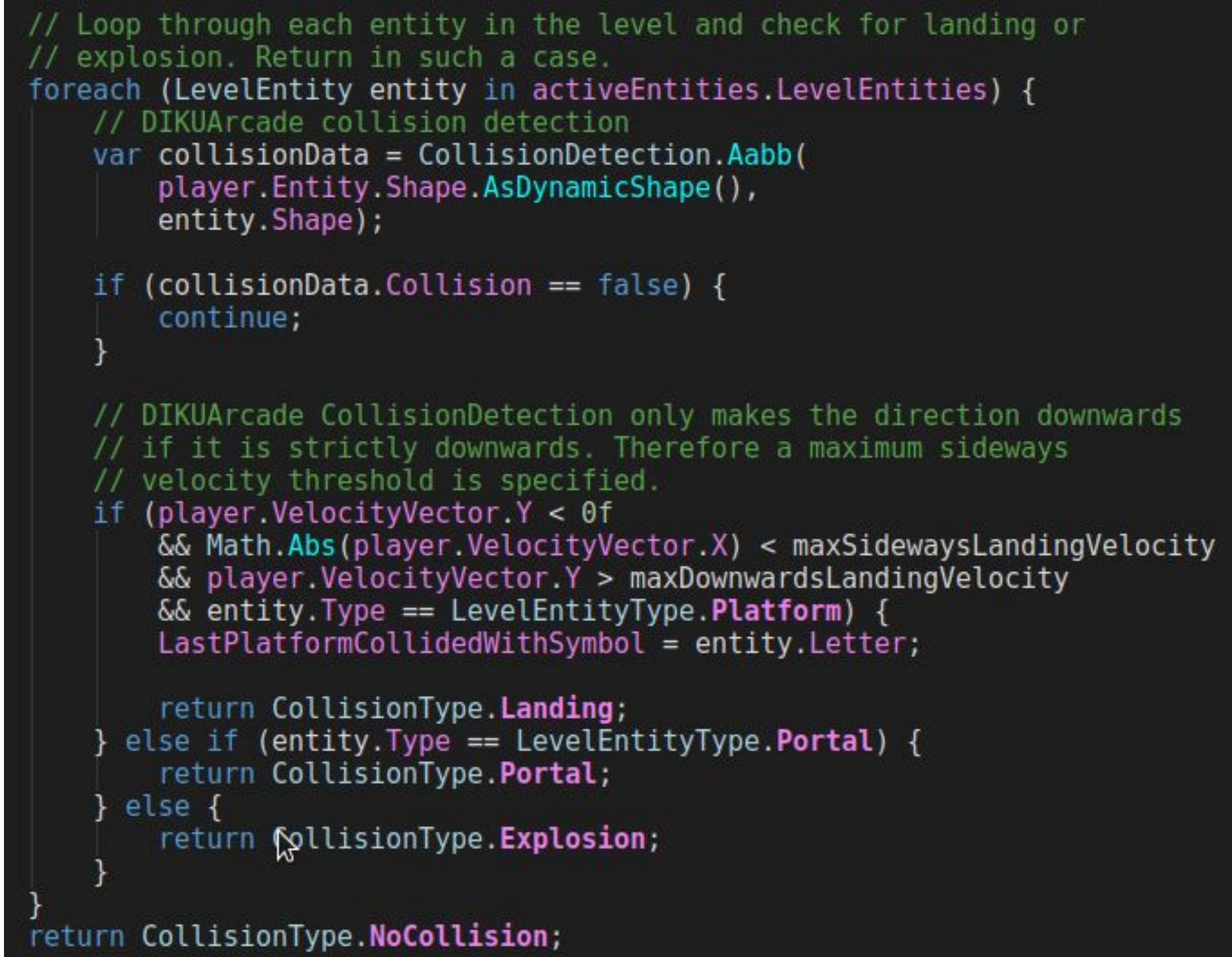
Collision (I)

REQ-6: The taxi should explode when hitting obstacles. When the explosion is over, the system shouldreturn to the main menu.
And REQ-7 (landing), REQ-8 (portals), REQ-10 (customer contact)


PlayerCollisionDetection
-activeEntites : ActiveEntities
+PlayerCollisionDetection(ActiveEntities)
+GetLatestCustomerCollidedWith() : Customer
+GetLatestPlatformCollidedWith() : char
-ValidCustomerCollision() : bool
+GetCollisionState() : CollisionType
<<enumeration>>
CollisionType
NoCollision
Explosion
Landing
Portal
Customer
ActiveEntities
+Level : LevelStatesEnum
+LevelEntites : EntityContainer<LevelEntity>
+Customers : EntityContainer<Customer>
+Player : IPlayer
0..*
Customer
0..*
LevelEntity
-type : LevelEntityType
-letter : char
DIKUArcade Entity
<<enumeration>>
LevelEntityType
Obstacle
Platform
Portal
Customer

Powered ByVisual Paradigm Community Edition

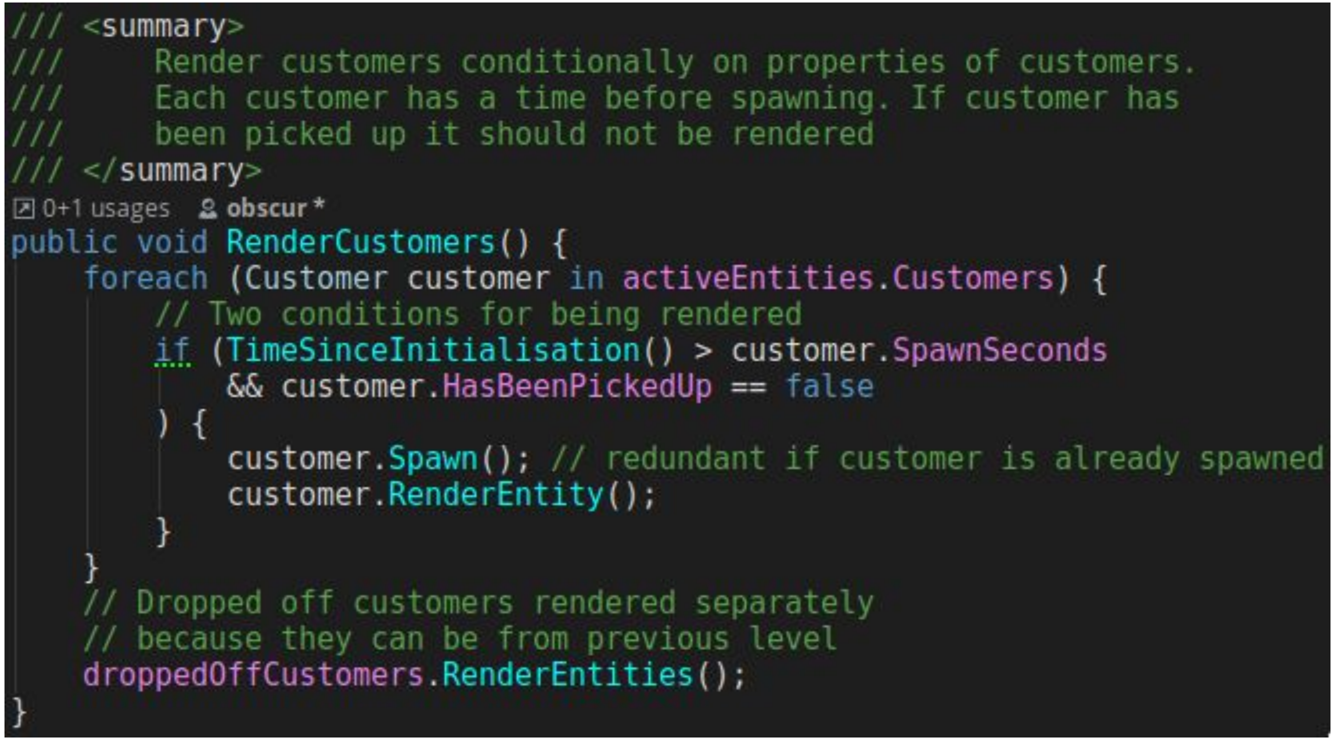
Collision (II)



```
// Loop through each entity in the level and check for landing or
// explosion. Return in such a case.
foreach (LevelEntity entity in activeEntities.LevelEntities) {
    // DIKUArcade collision detection
    var collisionData = CollisionDetection.Aabb(
        player.Entity.Shape.AsDynamicShape(),
        entity.Shape);

    if (collisionData.Collision == false) {
        continue;
    }

    // DIKUArcade CollisionDetection only makes the direction downwards
    // if it is strictly downwards. Therefore a maximum sideways
    // velocity threshold is specified.
    if (player.VelocityVector.Y < 0f
        && Math.Abs(player.VelocityVector.X) < maxSidewaysLandingVelocity
        && player.VelocityVector.Y > maxDownwardsLandingVelocity
        && entity.Type == LevelEntityType.Platform) {
        LastPlatformCollidedWithSymbol = entity.Letter;

        return CollisionType.Landing;
    } else if (entity.Type == LevelEntityType.Portal) {
        return CollisionType.Portal;
    } else {
        return CollisionType.Explosion;
    }
}
return CollisionType.NoCollision;
```

Spawning

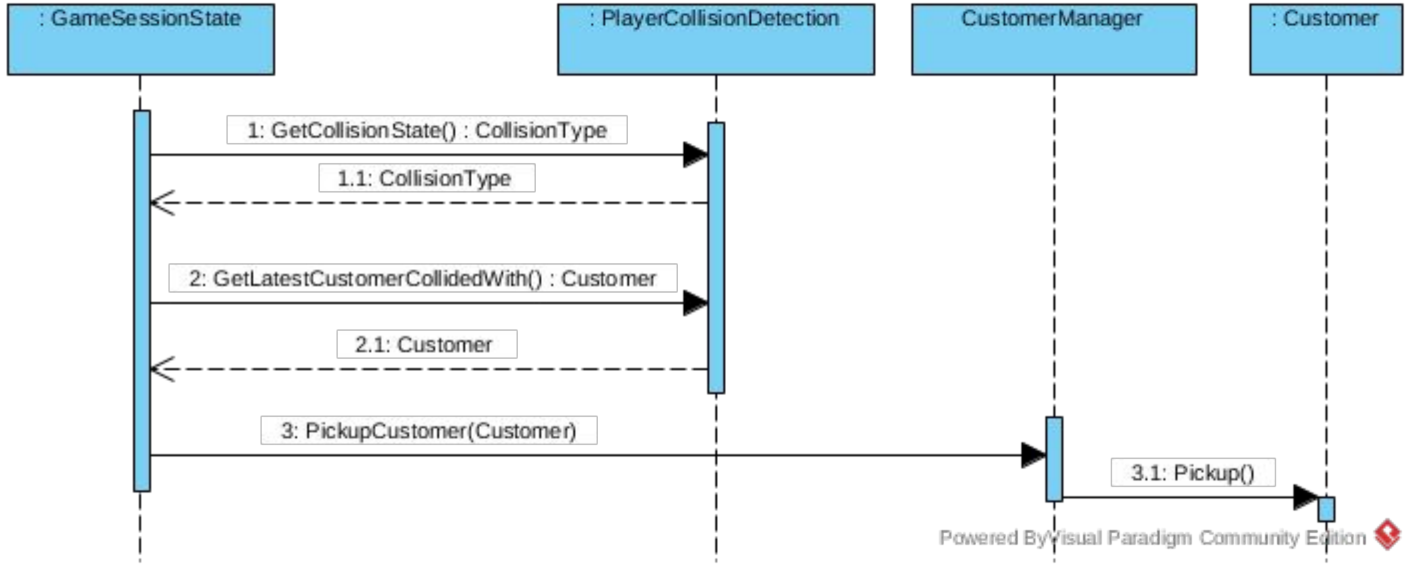
REQ-9: The customers corresponding to a given level should spawn on the given platforms after the number of seconds specified in the file has passed. [...]



```
/// <summary>
///     Render customers conditionally on properties of customers.
///     Each customer has a time before spawning. If customer has
///     been picked up it should not be rendered
/// </summary>
0+1 usages  obscur *
public void RenderCustomers() {
    foreach (Customer customer in activeEntities.Customers) {
        // Two conditions for being rendered
        if (TimeSinceInitialisation() > customer.SpawnSeconds
            && customer.HasBeenPickedUp == false
        ) {
            customer.Spawn(); // redundant if customer is already spawned
            customer.RenderEntity();
        }
    }
    // Dropped off customers rendered separately
    // because they can be from previous level
    droppedOffCustomers.RenderEntities();
}
```

Customer pickup

REQ-10: When the taxi collides with a customer, the customer should disappear from the screen.
and REQ-11



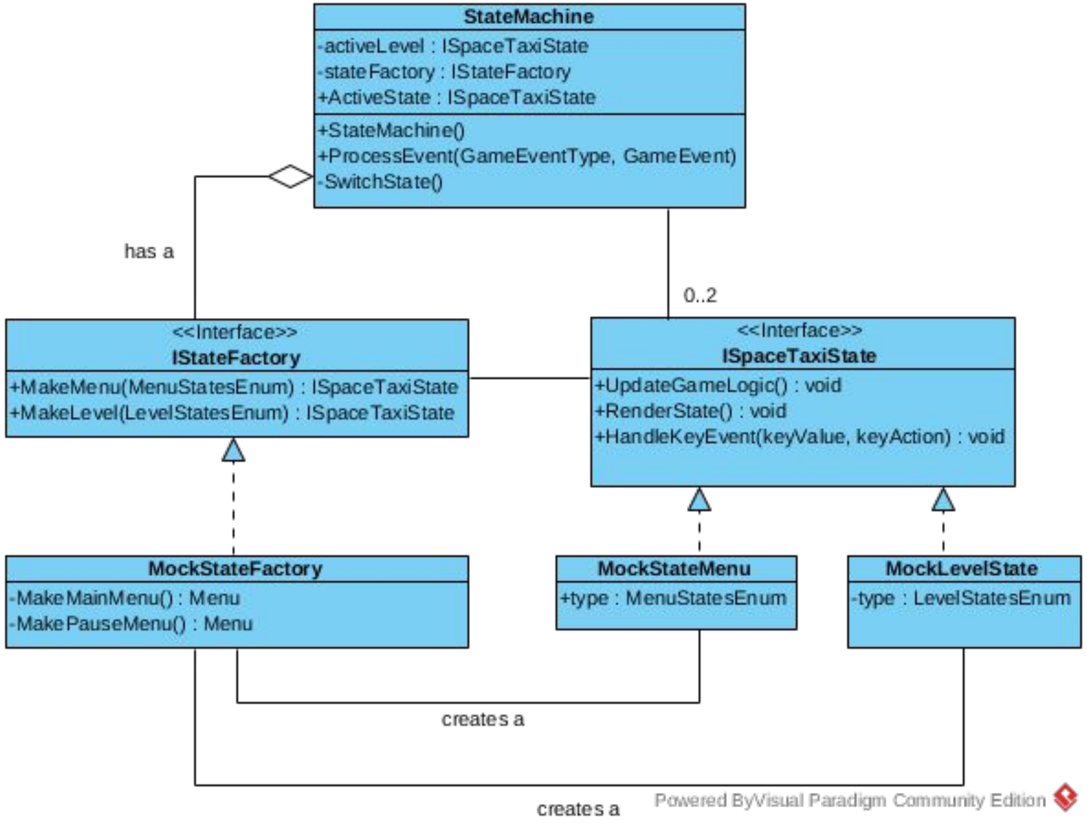
: GameSessionState
: PlayerCollisionDetection
CustomerManager
: Customer
1: GetCollisionState() : CollisionType
1.1: CollisionType
2: GetLatestCustomerCollidedWith() : Customer
2.1: Customer
3: PickupCustomer(Customer)
3.1: Pickup()
Powered By Visual Paradigm Community Edition

Evaluation

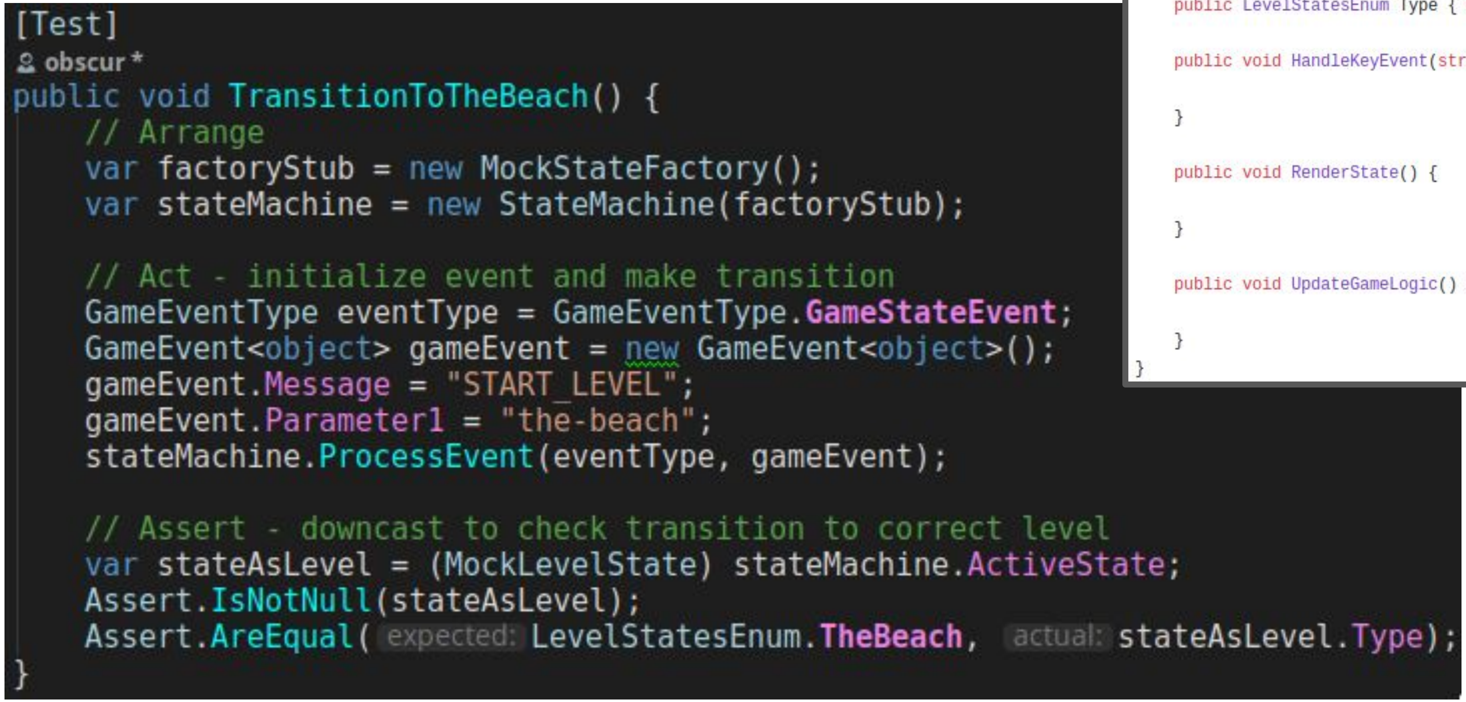
Requirements ~ concepts
...
REQ-4: IPlayer, IPhysicsCalculator
REQ-5: ITaxiAnimation
REQ-6: Explosions, IPlayerCollisionDetection
...

- Verification
- Validation
  - Units: StateMachine, EnumToStringTransformer, PhysicsCalculator
  - Integration testing
  - System testing

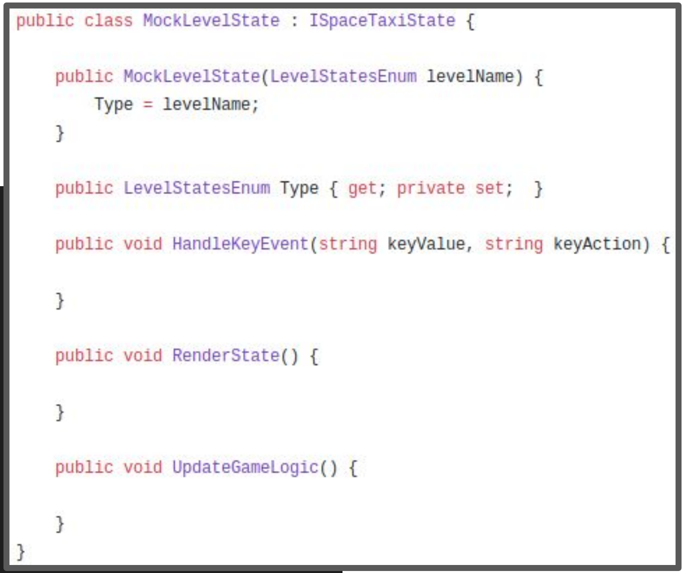
State machine (I)


StateMachine
-activeLevel : ISpaceTaxiState
-stateFactory : IStateFactory
+ActiveState : ISpaceTaxiState
+StateMachine()
+ProcessEvent(GameEventType, GameEvent)
-SwitchState()
has a
0..2
<<Interface>>
IStateFactory
+MakeMenu(MenuStatesEnum) : ISpaceTaxiState
+MakeLevel(LevelStatesEnum) : ISpaceTaxiState
<<Interface>>
ISpaceTaxiState
+UpdateGameLogic() : void
+RenderState() : void
+HandleKeyEvent(keyValue, keyAction) : void
MockStateFactory
-MakeMainMenu() : Menu
-MakePauseMenu() : Menu
MockStateMenu
+type : MenuStatesEnum
MockLevelState
-type : LevelStatesEnum
creates a
creates a
Powered By Visual Paradigm Community Edition

State machine (II)



```
[Test]
obscur *
public void TransitionToTheBeach() {
    // Arrange
    var factoryStub = new MockStateFactory();
    var stateMachine = new StateMachine(factoryStub);

    // Act - initialize event and make transition
    GameEventType eventType = GameEventType.GameStateEvent;
    GameEvent<object> gameEvent = new GameEvent<object>();
    gameEvent.Message = "START_LEVEL";
    gameEvent.Parameter1 = "the-beach";
    stateMachine.ProcessEvent(eventType, gameEvent);

    // Assert - downcast to check transition to correct level
    var stateAsLevel = (MockLevelState) stateMachine.ActiveState;
    Assert.IsNotNull(stateAsLevel);
    Assert.AreEqual( expected: LevelStatesEnum.TheBeach,  actual: stateAsLevel.Type);
}
```



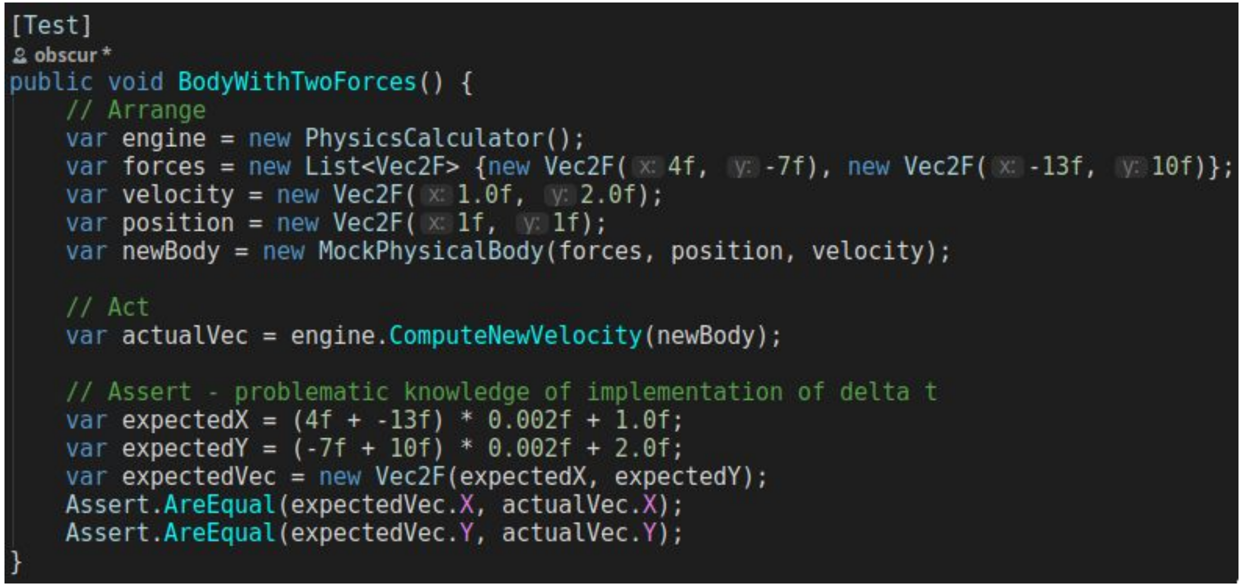
```
public class MockLevelState : ISpaceTaxiState {

    public MockLevelState(LevelStatesEnum levelName) {
        Type = levelName;
    }

    public LevelStatesEnum Type { get; private set;  }

    public void HandleKeyEvent(string keyValue, string keyAction) {

    }

    public void RenderState() {

    }

    public void UpdateGameLogic() {

    }
}
```

EnumStringTransformer

- Mapping, functional testing
- Equivalence classes

PhysicsCalculator

- Abstraction, well-defined function
- Knowledge of implementation
- Structured path coverage k=2
- Integration with Player



```
[Test]
obscur *
public void BodyWithTwoForces() {
    // Arrange
    var engine = new PhysicsCalculator();
    var forces = new List<Vec2F> {new Vec2F( x: 4f,  y: -7f), new Vec2F( x: -13f,  y: 10f)};
    var velocity = new Vec2F( x: 1.0f,  y: 2.0f);
    var position = new Vec2F( x: 1f,  y: 1f);
    var newBody = new MockPhysicalBody(forces, position, velocity);

    // Act
    var actualVec = engine.ComputeNewVelocity(newBody);

    // Assert - problematic knowledge of implementation of delta t
    var expectedX = (4f + -13f) * 0.002f + 1.0f;
    var expectedY = (-7f + 10f) * 0.002f + 2.0f;
    var expectedVec = new Vec2F(expectedX, expectedY);
    Assert.AreEqual(expectedVec.X, actualVec.X);
    Assert.AreEqual(expectedVec.Y, actualVec.Y);
}
```

Sample use case

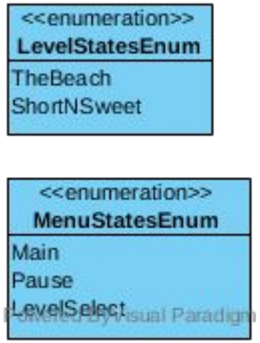
Use case UC-1	Landing (in The Beach?)
Related requirements	REQ-4 and REQ-7
Initiating actor	Player
Actor’s goal:	Landing on the platform (in some cases to drop off a customer?)
Participating actors:	Taxi, platform
Pre-conditions:	Game is running, the taxi has a low downwards velocity, the direction of the taxi is downwards, the taxi is in contact with the platform.
Post-conditions:	The taxi does not move.
Flow of events for main success scenario	
1. ->	User presses activates up-boosters often enough to still fall but without without gaining high downwards velocity until contacting platform.
2. <-	The system sets the velocity of the taxi to zero.

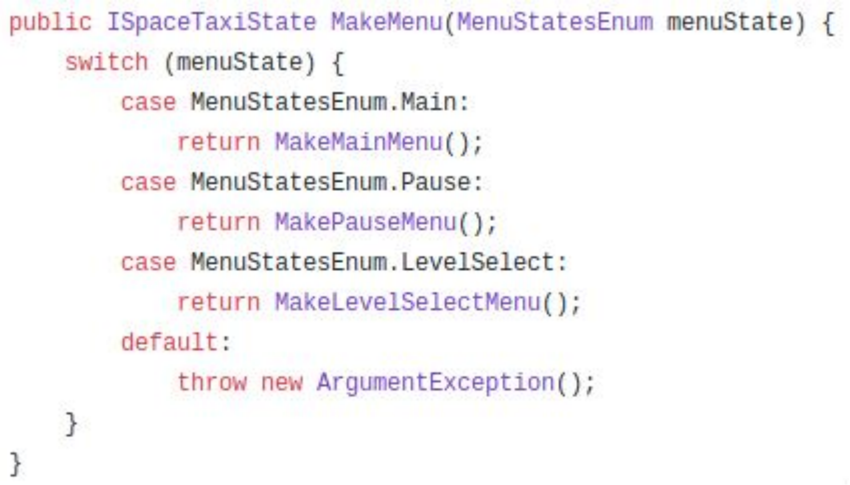
Extra talking points:

- Quality
- Tools: Rider, NUnit, git, Visual Paradigm
- Frameworks and libraries
- Agility

Adaptiveness, enums, defensive

- Enums
- Configuration
- Exceptions, defensive programming

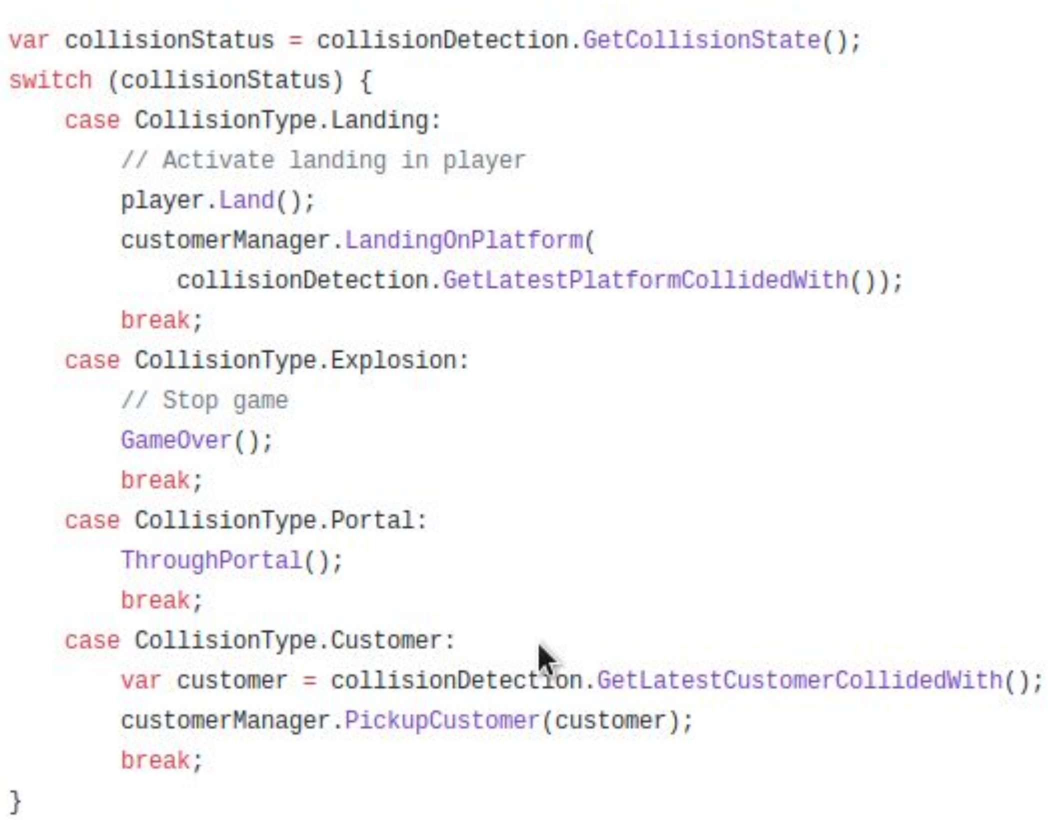



```
public ISpaceTaxiState MakeMenu(MenuStatesEnum menuState) {
    switch (menuState) {
        case MenuStatesEnum.Main:
            return MakeMainMenu();
        case MenuStatesEnum.Pause:
            return MakePauseMenu();
        case MenuStatesEnum.LevelSelect:
            return MakeLevelSelectMenu();
        default:
            throw new ArgumentException();
    }
}
```

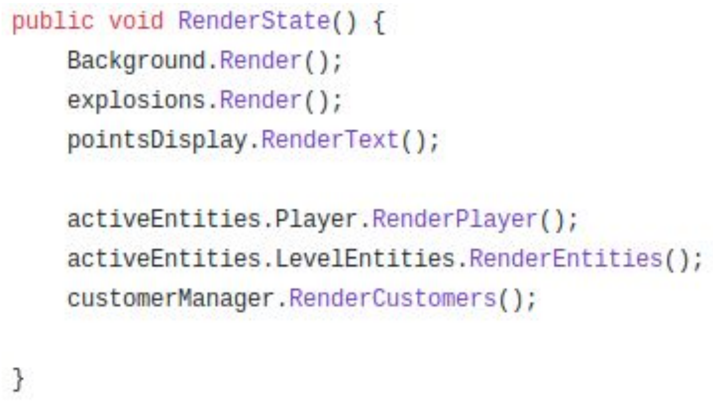
Other code

GameSessionState code

composite?

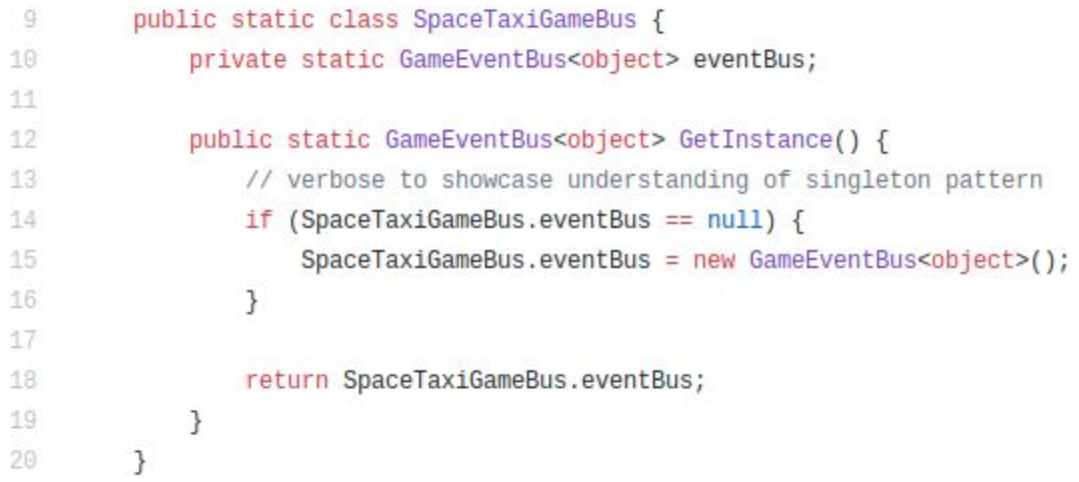


```
var collisionStatus = collisionDetection.GetCollisionState();
switch (collisionStatus) {
    case CollisionType.Landing:
        // Activate landing in player
        player.Land();
        customerManager.LandingOnPlatform(
            collisionDetection.GetLatestPlatformCollidedWith());
        break;
    case CollisionType.Explosion:
        // Stop game
        GameOver();
        break;
    case CollisionType.Portal:
        ThroughPortal();
        break;
    case CollisionType.Customer:
        var customer = collisionDetection.GetLatestCustomerCollidedWith();
        customerManager.PickupCustomer(customer);
        break;
}
```



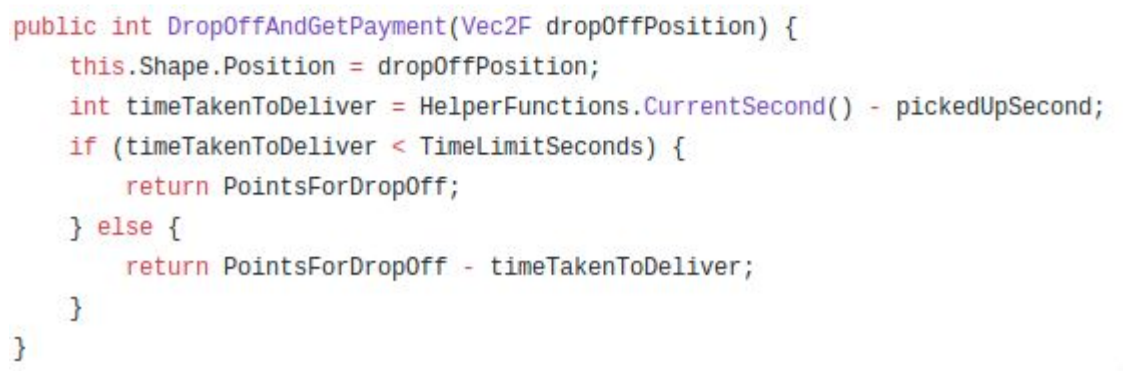
```
public void RenderState() {
    Background.Render();
    explosions.Render();
    pointsDisplay.RenderText();

    activeEntities.Player.RenderPlayer();
    activeEntities.LevelEntities.RenderEntities();
    customerManager.RenderCustomers();

}
```

SpaceTaxiGameBus code



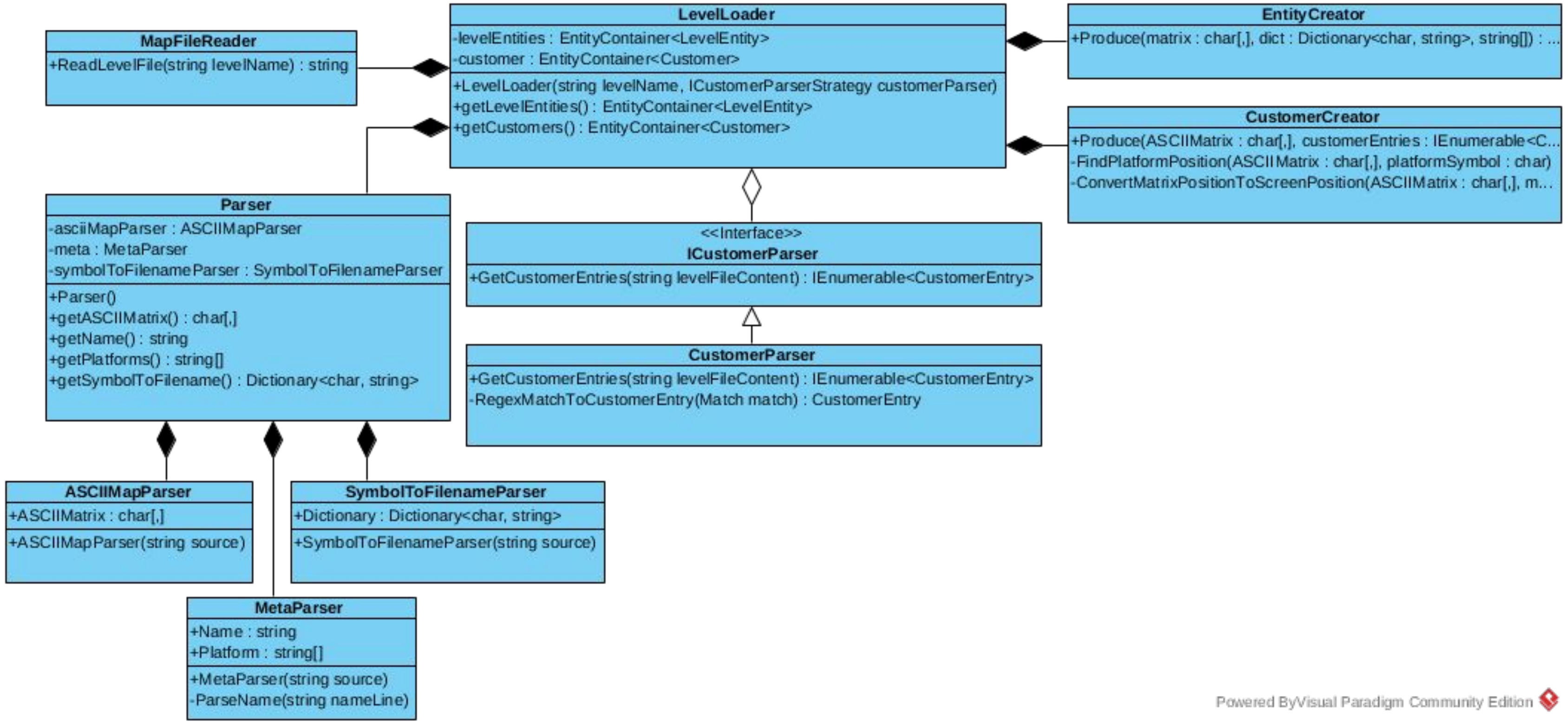
```
    public static class SpaceTaxiGameBus {
        private static GameEventBus<object> eventBus;

        public static GameEventBus<object> GetInstance() {
            // verbose to showcase understanding of singleton pattern
            if (SpaceTaxiGameBus.eventBus == null) {
                SpaceTaxiGameBus.eventBus = new GameEventBus<object>();
            }

            return SpaceTaxiGameBus.eventBus;
        }
    }
```

Customer code



```
public int DropOffAndGetPayment(Vec2F dropOffPosition) {
    this.Shape.Position = dropOffPosition;
    int timeTakenToDeliver = HelperFunctions.CurrentSecond() - pickedUpSecond;
    if (timeTakenToDeliver < TimeLimitSeconds) {
        return PointsForDropOff;
    } else {
        return PointsForDropOff - timeTakenToDeliver;
    }
}
```

Level loading


MapFileReader
+ReadLevelFile(string levelName) : string
LevelLoader
-levelEntities : EntityContainer<LevelEntity>
-customer : EntityContainer<Customer>
+LevelLoader(string levelName, ICustomerParserStrategy customerParser)
+getLevelEntities() : EntityContainer<LevelEntity>
+getCustomers() : EntityContainer<Customer>
EntityCreator
+Produce(matrix : char[,], dict : Dictionary<char, string>, string[]) : ...
CustomerCreator
+Produce(ASCIIMatrix : char[,], customerEntries : IEnumerable<C...
-FindPlatformPosition(ASCIIMatrix : char[,], platformSymbol : char)
-ConvertMatrixPositionToScreenPosition(ASCIIMatrix : char[,], m...
Parser
-asciiMapParser : ASCIIMapParser
-meta : MetaParser
-symbolToFilenameParser : SymbolToFilenameParser
+Parser()
+getASCIIMatrix() : char[,]
+getName() : string
+getPlatforms() : string[]
+getSymbolToFilename() : Dictionary<char, string>
<<Interface>>
ICustomerParser
+GetCustomerEntries(string levelFileContent) : IEnumerable<CustomerEntry>
CustomerParser
+GetCustomerEntries(string levelFileContent) : IEnumerable<CustomerEntry>
-RegexMatchToCustomerEntry(Match match) : CustomerEntry
ASCIIMapParser
+ASCIIMatrix : char[,]
+ASCIIMapParser(string source)
SymbolToFilenameParser
+Dictionary : Dictionary<char, string>
+SymbolToFilenameParser(string source)
MetaParser
+Name : string
+Platform : string[]
+MetaParser(string source)
-ParseName(string nameLine)