System Design Document for GTFA

Samuel Utbult, Anton Levholm, John Andersson, Marcus Eliasson

26 May 2016

Contents

1 Introduction

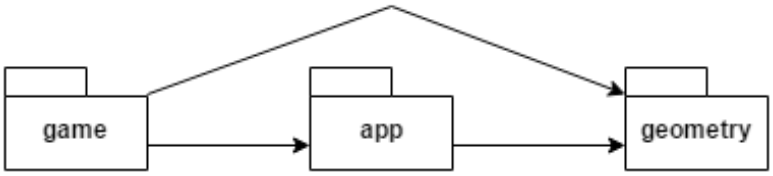
1.1 Design goals

One of the design goals of the application is aimed to separate game logic from library connection as much as possible. This will make it possible to switch library without interfering with the game logic. The parts of the program that handles triangle management is also supposed to be loosely connected to the game logic, as it then can be reused in other games. The game logic should have good structure that would make it easy to add or remove any components of the game.

2 System design

2.1 Software decomposition

2.1.1 Layering



The application is divided into three layers; the game, application and geometry layers.

- The game layer contains the game logic.
- The application layer provides an interface between the game logic and the IO library.
- The geometry layer contains logic for manageing positions and processing triangle collision detection.

The application layer depends on the geometry layer and the game layer depends on both of them.

2.1.2 Decomposition into subsystems

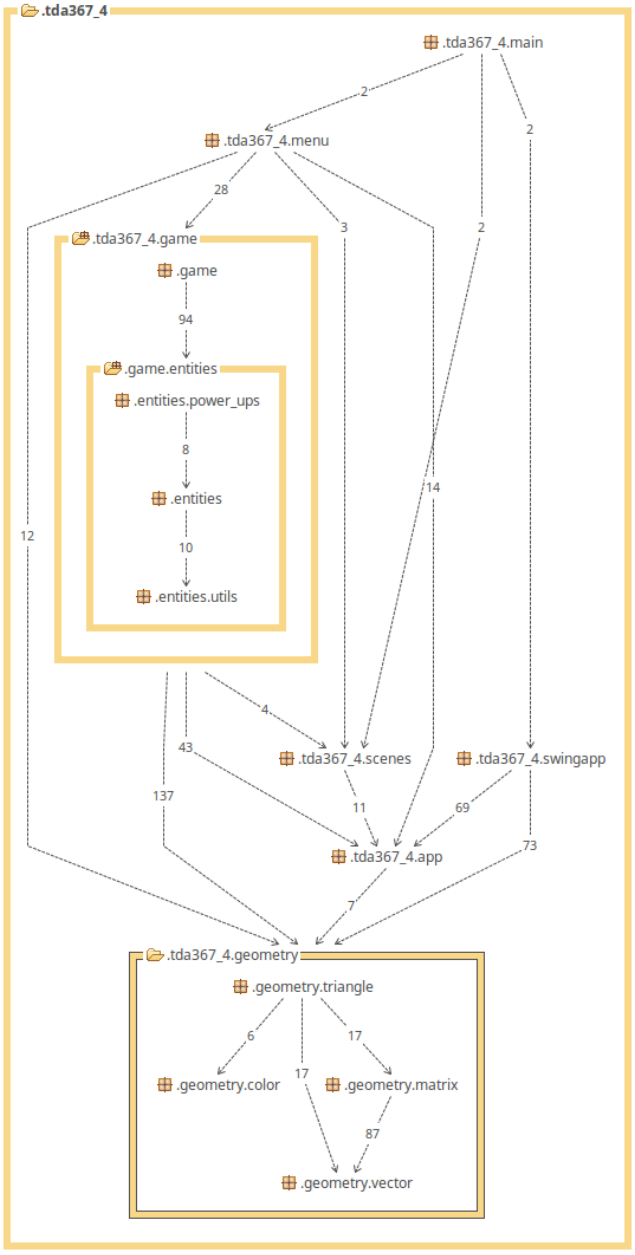
The Application layer is the main subsystem of the game. The application is presented to the game logic as a number of interfaces. These interfaces are then implemented by a class that handles the connection to Swing. The benefit of this practice is that the game logic becomes library independent and that the application implementation can easely be switched to another implementation using a different library, such as OpenGL.

2.1.3 Game logic

The game logic is structured according to the *Composition over inheritance* pattern (Wikipedia, 2016). In short, systems using the pattern should have a number of interfaces that each symbolizes a property with a number of methods that further specifies the property. Then there is a number of classes that each symbolizes the different kinds of entities that the game contains. These classes inherits from the interfaces that represents the desired properties.

These classes are managed by a root class that holds references to all game entities and calls the appropriate methods of the interfaces at the right times.

2.1.4 Dependency analysis



The java packages closely follows the previously described layers of the application. The game and geometry layer is represented by two packages, with inner

packages that further divides functionality. The application layer is divided into two packages, the 'app' package and the 'swingapp' package, where app contains the application interfaces and swingapp contains the swing implementation of the application layer.

Notice that game does not depend on swingapp and that main depends on both game and swingapp, as main's sole purpose is to launch the application. An additional package exists called 'menu' which contains the game menu.

2.2 Concurrency issues

The Swing implementation creates a second thread that processes all the logic. The single exception is that keyboard events arrive to the application in a separate thread that is saved for later usage. Therefore, the methods that accesses the same data but are executed on different threads are synchronized.

2.3 Persistent data management

The only information that is stored by the game permanently is the user's high scores. These are stored on the user's hard drive in a text file. Each row contains an integer representing the score of one playthrough.

2.4 Access control and security

There is no networking in the application and it does not manage vital parts of the user's computer, for instance the operating system. Therefore, there will not be any larger problems with access control and security.

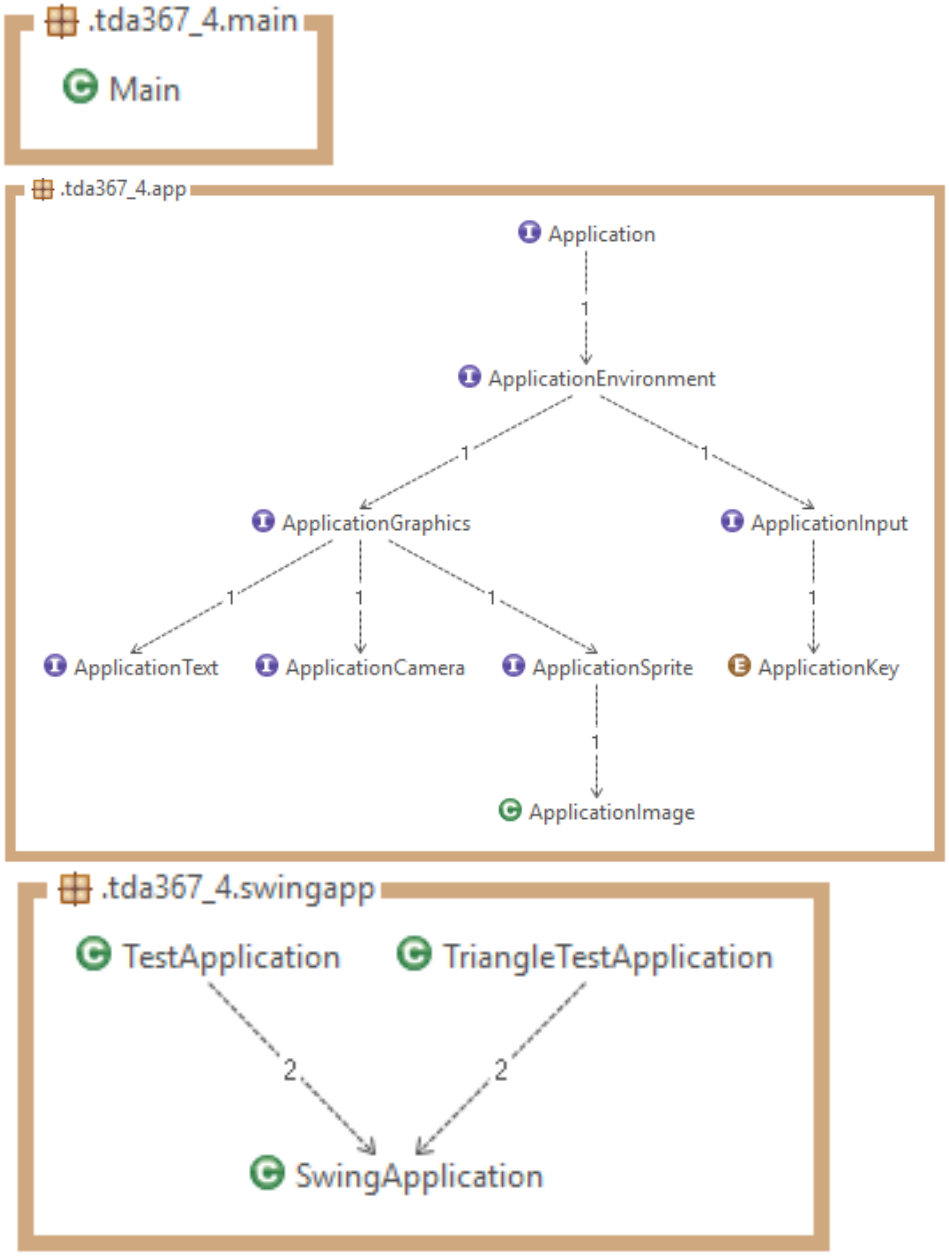
2.5 Boundary conditions

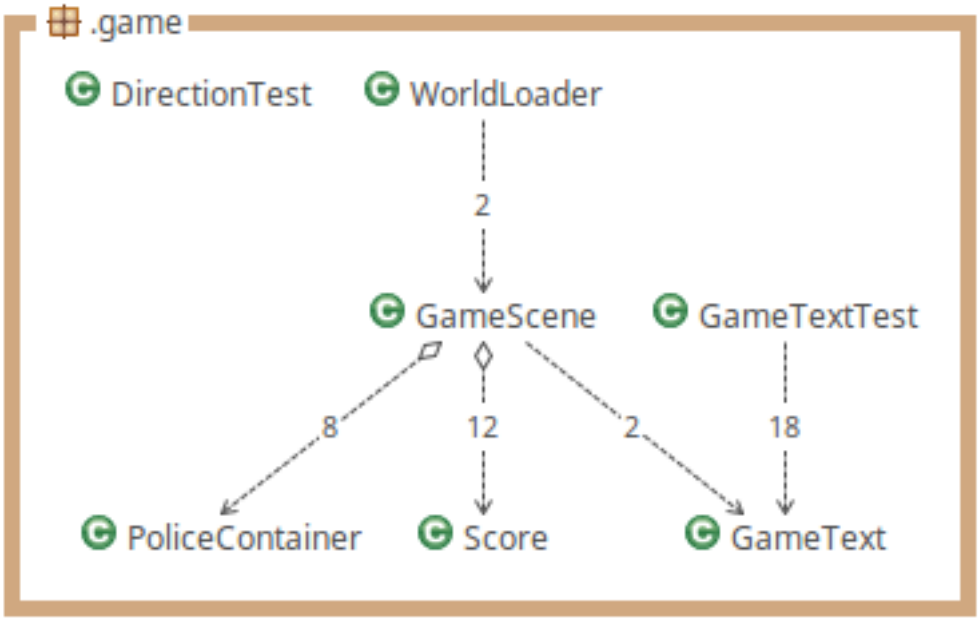
The application is launched and exited as a normal desktop application and does not manage any other processes. Therefore, this will not cause any issues.

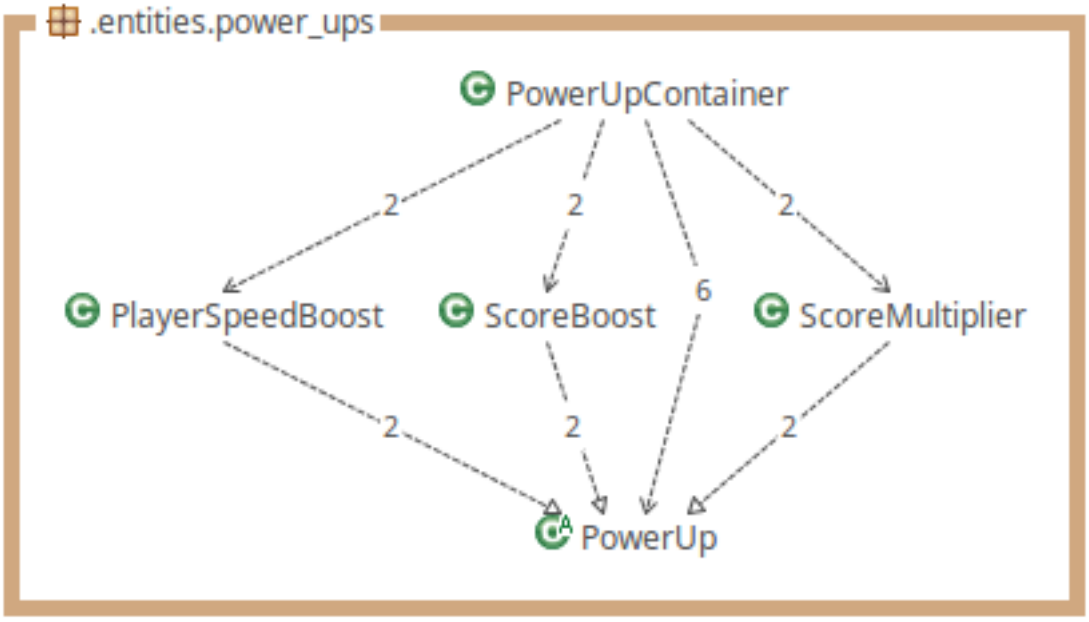
3 References

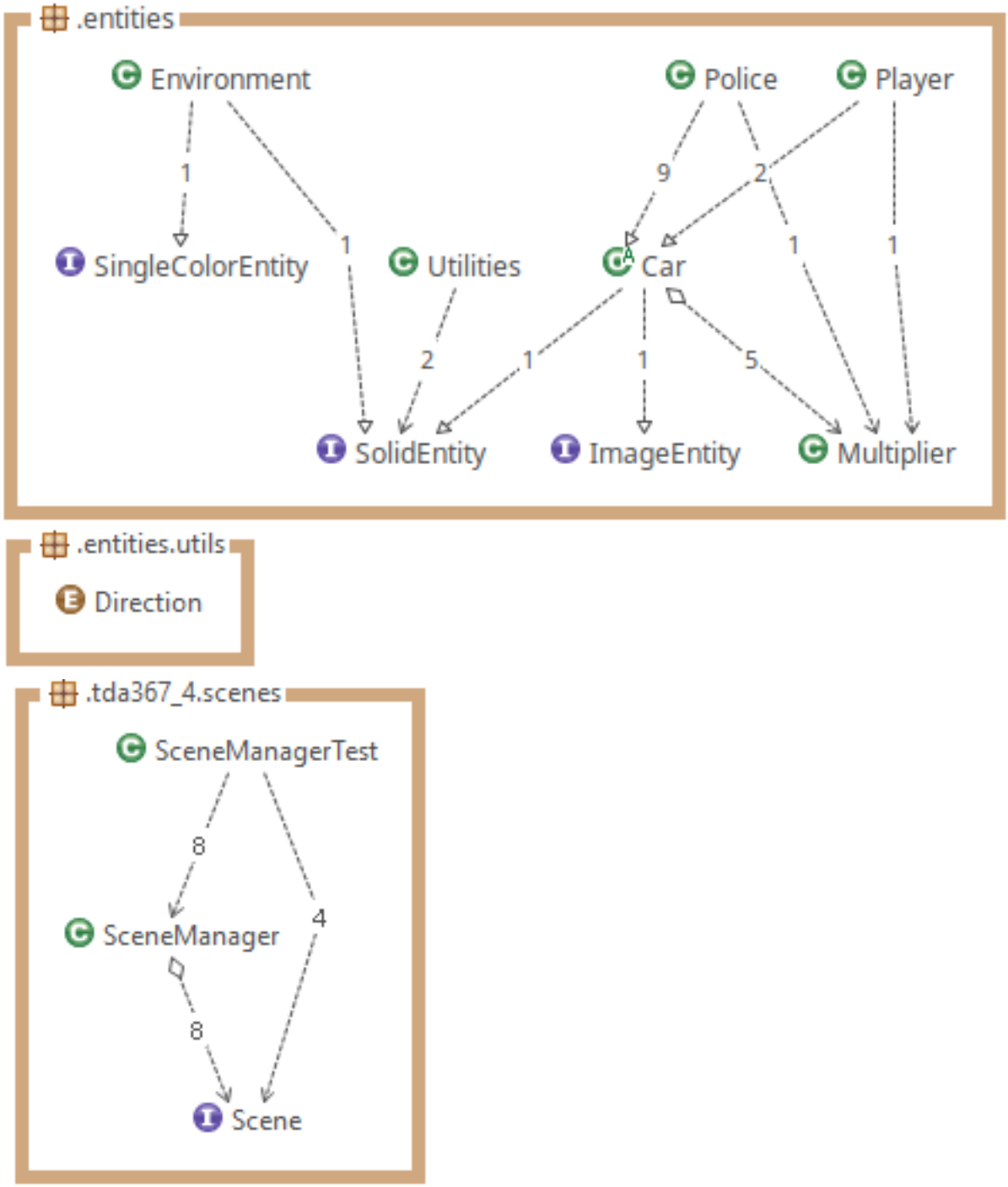
Wikipedia. (2016). *Composition over inheritance.* Retrieved 2016-05-18, from `https://en.wikipedia.org/wiki/Composition_over_inheritance`

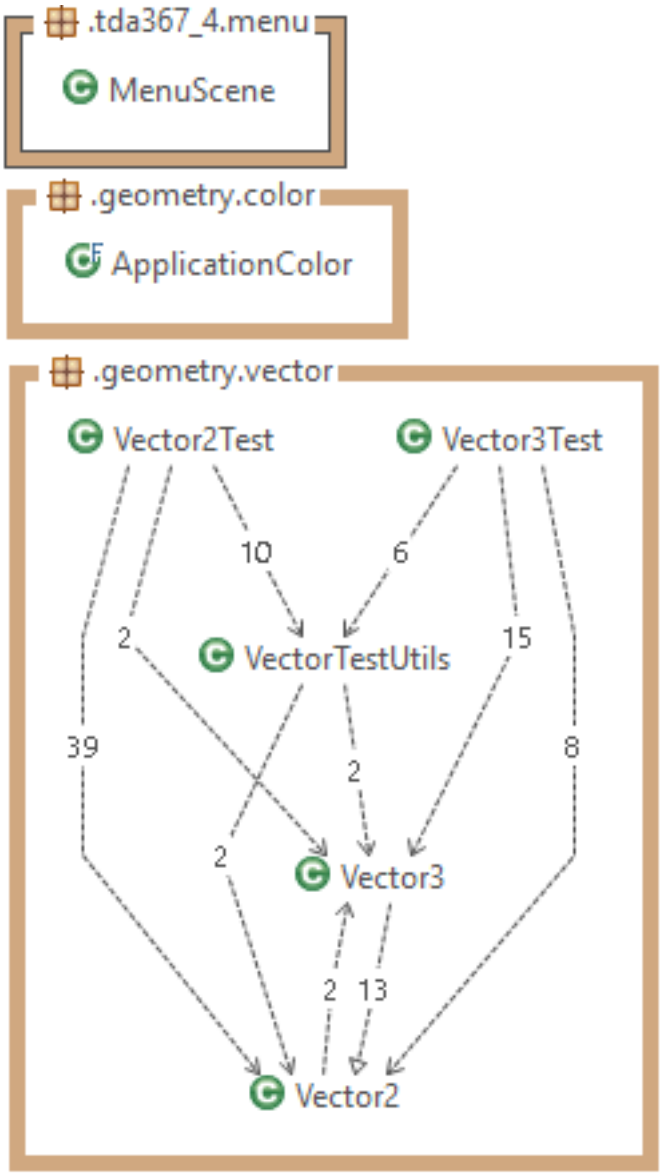
Appendix

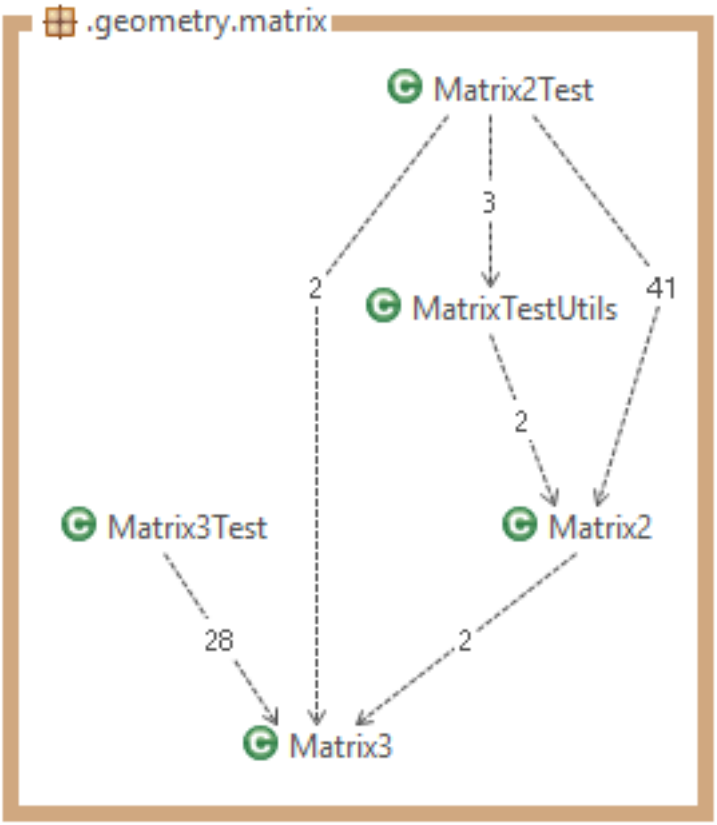

.tda367_4.main
Main
.tda367_4.app
Application
1
ApplicationEnvironment
1
1
ApplicationGraphics
ApplicationInput
1
1
1
1
ApplicationText
ApplicationCamera
ApplicationSprite
ApplicationKey
1
ApplicationImage
.tda367_4.swingapp
TestApplication
TriangleTestApplication
2
2
SwingApplication

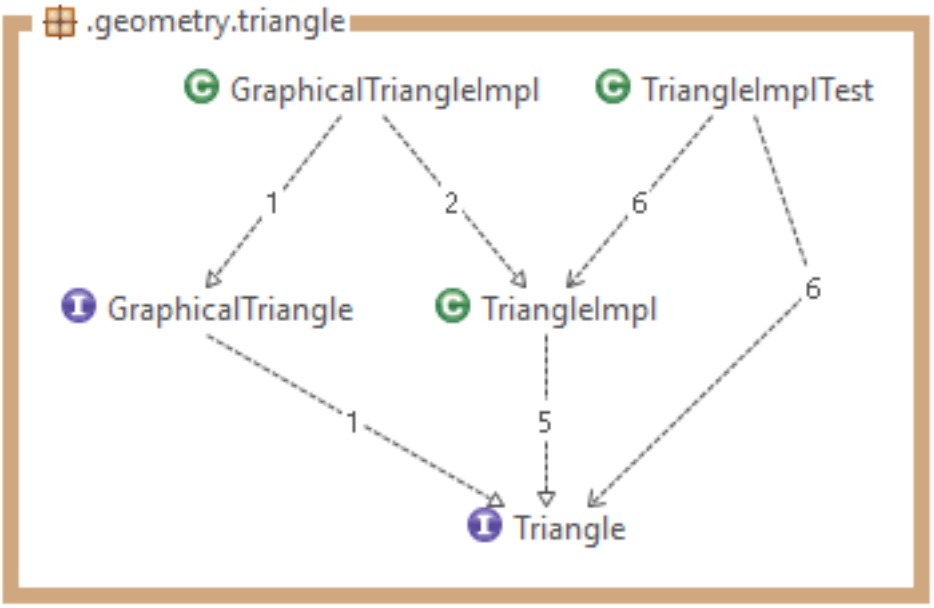

.game
DirectionTest
WorldLoader
2
GameScene
GameTextTest
8
12
2
18
PoliceContainer
Score
GameText


.entities.power_ups
PowerUpContainer
2
2
2
6
PlayerSpeedBoost
ScoreBoost
ScoreMultiplier
2
2
2
PowerUp


.entities
Environment
Police
Player
1
9
2
SingleColorEntity
1
Utilities
Car
1
1
2
1
1
5
SolidEntity
ImageEntity
Multiplier
.entities.utils
Direction
.tda367_4.scenes
SceneManagerTest
8
4
SceneManager
8
Scene


.tda367_4.menu
MenuScene
.geometry.color
ApplicationColor
.geometry.vector
Vector2Test
Vector3Test
10
6
2
15
VectorTestUtils
39
2
8
2
Vector3
2
13
Vector2


.geometry.matrix
Matrix2Test
3
2
41
MatrixTestUtils
2
Matrix3Test
Matrix2
28
2
Matrix3


.geometry.triangle
GraphicalTriangleImpl
TriangleImplTest
1
2
6
6
GraphicalTriangle
TriangleImpl
1
5
Triangle