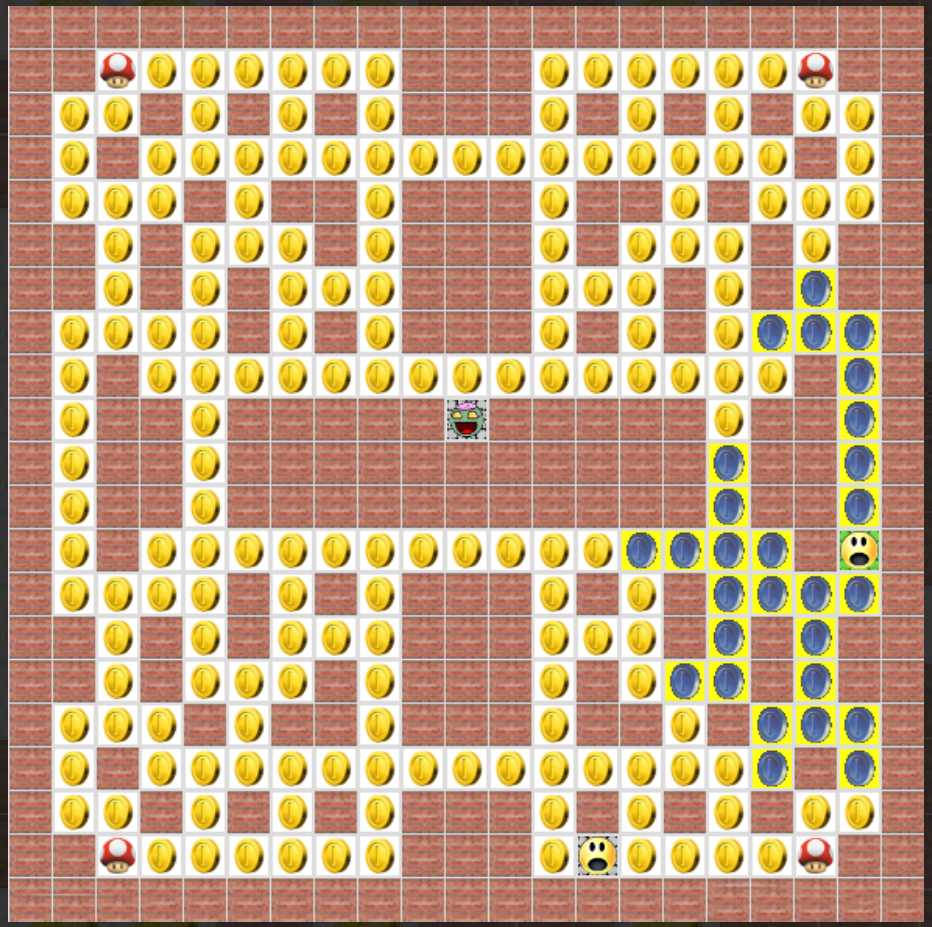
TacZombie

(a tactical Pacman clone)



Developed by Lars Eckervogt, Manuel Hieke and Stefan Junker for the course “Moderne Programmiersprachen” in MSI1 in WS13/14 at HTWG Konstanz

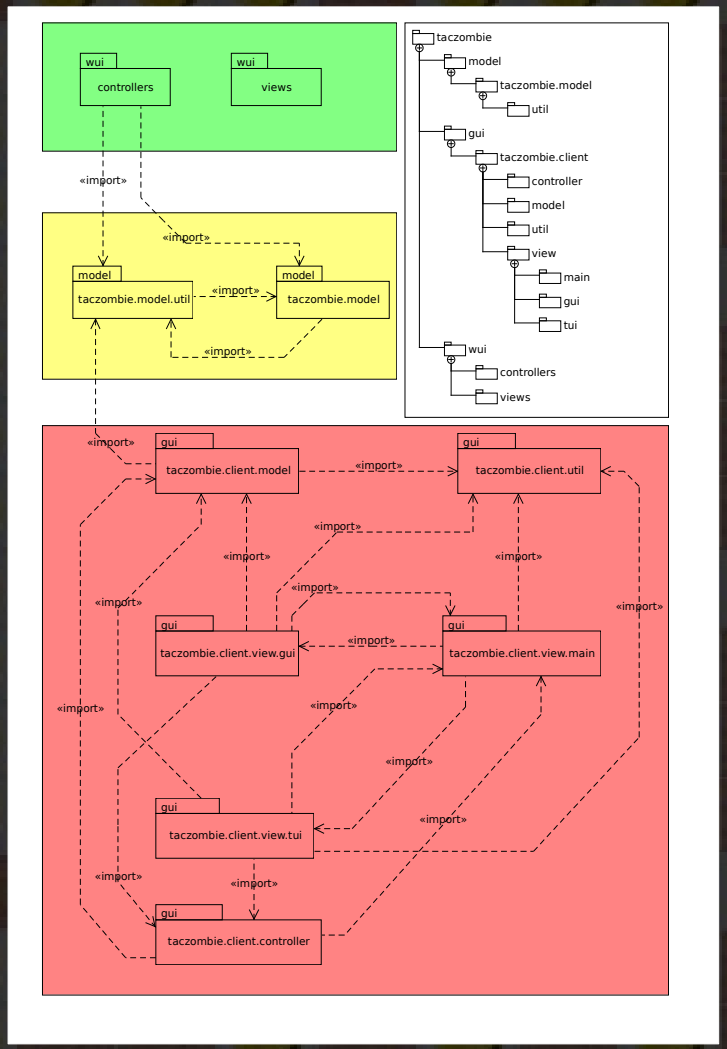
Content

- Game Principle
- Worth a Word
- Tests
- Demo
- Summary

Game Principle

- Round-based
- Two Players
  - **One** Human: can have multiple tokens
  - **One** Zombie: can have multiple tokens
- Goal
  - Zombie: kill Human
  - Human: survive and collect all coins
- Map
  - Pacman-like

Structure


wui
controllers
wui
views
«import»
«import»
model
taczombie.model.util
«import»
model
taczombie.model
«import»
taczombie
model
taczombie.model
util
gui
taczombie.client
controller
model
util
view
main
gui
tui
wui
controllers
views
«import»
gui
taczombie.client.model
«import»
gui
taczombie.client.util
«import»
«import»
«import»
«import»
«import»
gui
taczombie.client.view.gui
«import»
gui
taczombie.client.view.main
«import»
«import»
«import»
«import»
gui
taczombie.client.view.tui
«import»
«import»
«import»
gui
taczombie.client.controller

Model

- Case classes
  - Easy access of subsets
- Build to be extensible
  - Multiple Players (Zombie/Humans)

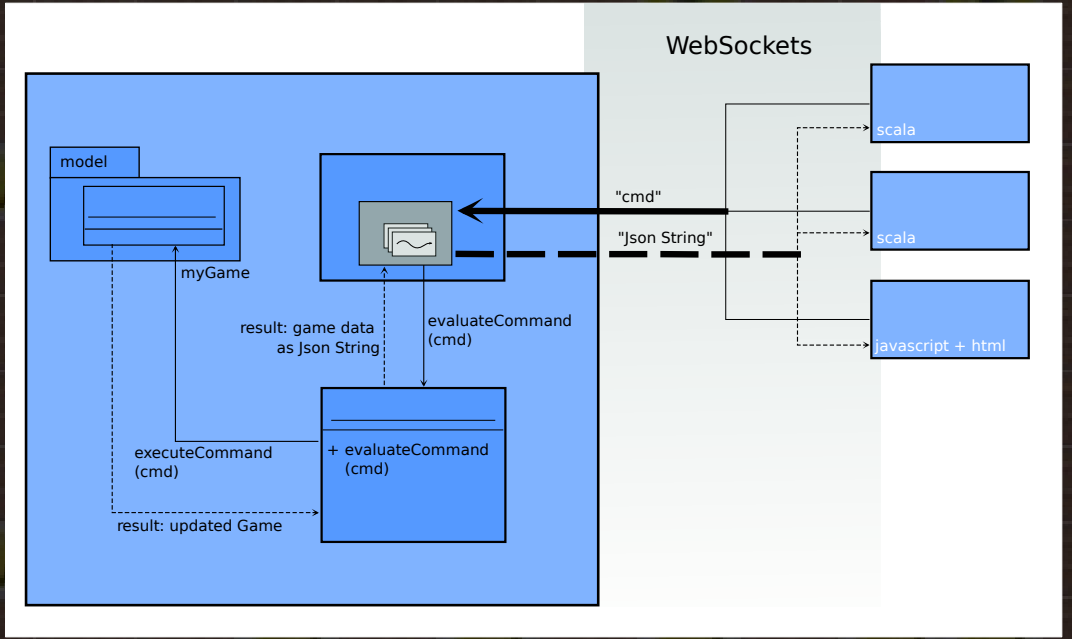
Views

- WUI
- TUI
- GUI

- Communication through web sockets
  - Library: just.ws (https://github.com/stasimus/just.ws)
- Transport format: jSON
  - Library: io.spray (https://github.com/spray/spray-json)

Controller

- Server-side: Play
  - Manages web socket connections
  - Manages the game
  - Handles incoming client requests
  - Servers game data to the clients
- Client-side:
  - GUI: Swing
  - WUI: JavaScript/Play



Use of Traits

- Example: Logger

```
5  trait Logger {
6    object logger {
7
8      private var printOnGet : Boolean = false
9
10     private val data : ListBuffer[String] = ListBuffer[String]()
11
12     def clear = data.clear()
13
14     def +=(s : String, print : Boolean = false) = {
15       data += s
16       if(print) println(s)
17     }
18
19     def init(s : String, print : Boolean = false, printOnGet : Boolean = false) =  {
20       clear
21       this.printOnGet = printOnGet
22       +=(s, print)
23     }
```

Use of Traits

- Example: Logger

```
24
25      def get : List[String] = {
26        if(printOnGet) print
27        data.toList
28      }
29
30      def merge(l : Logger) = {
31        data.++=(l.logger.get)
32        l
33      }
34
35      def print = {
36        if(data.size > 0) {
37          println(data.apply(0))
38          for(s <- data.tail)
39            println("\t" + s)
40        } else  println("Empty logger")
41      }
42    }
43  }
```

Use of Dependency Injection

- Main Method GUI / TUI

```
17 object Main {
18   def main(args: Array[String]) {
19     var restart = true
20     val viewInjector = Guice.createInjector(new UiModule(args))
21
22     while (restart) {
23       val view = viewInjector.getInstance(classOf[IView])
24       view.open
25       restart = view.runBlocking
26     }
27   }
28 }
29
```

Use of Implicit Conversions

- Example: CoordinateHelper

```
3  import scala.collection.mutable.ListBuffer
4  import scala.language.implicitConversions
5  import taczombie.model.Game
6  import taczombie.model.GameFieldCell
7  import taczombie.model.GameField
8
9  object CoordinateHelper {
10
11   implicit def intIntTuple2Wrapper(tuple: (Int,Int)) =
12       new IntIntTuple2Helper(tuple)
13
14   class IntIntTuple2Helper(tuple : (Int,Int)) {
15     def leftOf : (Int,Int) = (tuple._1, tuple._2 - 1)
16     def rightOf : (Int,Int) = (tuple._1, tuple._2 + 1)
17     def aboveOf : (Int,Int) = (tuple._1 - 1, tuple._2)
18     def belowOf : (Int,Int) = (tuple._1 + 1, tuple._2)
19   }
20 }
```

Use of Higher Order Function

- Example: JsonHelperSpec

```
16   val -- = (a:Int,b:Int) => a - b
17   val ++ = (a:Int,b:Int) => a + b
18
19   private def upperLeftFor5StepsInBothDimensions(coord : (Int,Int))
20     (y : Int, fn1 : (Int,Int) => Int)
21     (x : Int, fn2 : (Int,Int) => Int) = {
22     for {
23           i <- 0 until 5
24           j <- 0 until 5
25     } yield coord.isUpperLeftOf(fn1(y,i),fn2(x,j))
26   }.toList
 :
60       upperLeftFor5StepsInBothDimensions(testCoord1)(10, --)(11, ++)
61         .exists(_ == true) must be_!=(true)
```

Tests

- 200 fully automated Tests
- Code Coverage scct
  - Model 100%
  - GUI / TUI: Utils 100%
  - Controller 76%
- Coverage Report as html

Generate Standalone Game

- Server/WUI
  - sbt “project wui” dist
  - Generates zip file
  - Extract and execute Skript wui in bin/ to run server
- GUI/TUI
  - sbt assembly
  - Generates TacZombieClient.jar
  - Execute java -jar TacZombieClient.jar for GUI
  - Add parameter “tui” for TUI

Demo

- Lean back and enjoy :-)

Summary

- GUI & TUI & WUI usable
  - at the ***same*** time
  - on the ***same*** model
- Model fully automated tested
- Controller automated tested
- SBT Multi-project

- Scala is awesome!
- Swing is ok
- ScalaFx lacks of support and examples