Course manual

Academy Creative Technology

Game Technology and Producing

Course: 1.4 Game Architecture

Version	2016.001
Quarter	1.4
Date of publication	20-05-2017

Courseleader:	Eelco Jannink
	e.h.a.jannink@saxion.nl
Tutors	Hans Wichman
	Eelco Jannink

Contents

1 General Discription

In Game Architecture we are going to abstract a bit and focus on the design of game software including engines. Coding is for proof of design here.

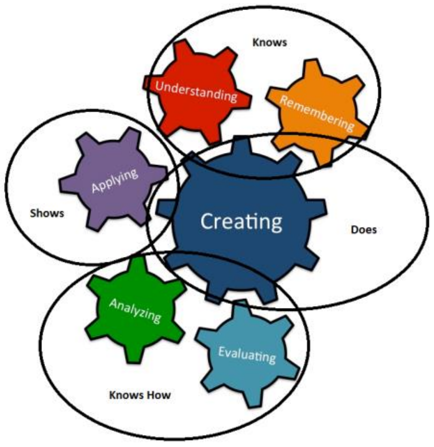
Coding is a basic skill needed to develop games. But any coding fails if the software design has flaws. This effect increases with larges code bases, bigger development teams and longer project runtime. But even smaller projects benefit from good design. It helps to avoid needless time loss and complicated bugs.

So design is important. This requires a language to express the design decisions. The Unified Modelling Language, UML, is our friend here. It is widely used in the industry and has loads of different schematics.

We'll crack this nut by learning to visualize software design using UML diagrams and study and apply Design Patterns. The exam has two parts, the written exam, aiming at the required knowledge and the assignment, demonstrating the actual use of both patterns and diagrams.
The written exam and the assignment both weigh 0.5 of the final grade.

Details on the schedule can be found at roosters.saxion.nl whilst the schedule for written exams is mysaxion.nl .
All information concerning the course is to be found blackboard.saxion.nl .

Learning goals



Knows How

- compares the structure of several games/engines/tooling
- compares the code quality and use of conventions of several games/engines with tooling

Shows How

- uses UML diagrams to visualize game structures and workings
- designs a test-plan for game development (software)
- demonstrates how to reverse engineer an existing game/engine with tooling

2 Organisation

Format of the course
We have five weeks to cover this topic. It requires an investment of 3 ects /hours.
We use the C# Windows Forms based simple GaGame as a vehicle.

Week 4:

Lecture	Intro, OO-Design, Patterns, Assertions
Lab	Apply Patterns to GaGame : Gameloop, Update, Component, State and Command. Setup services matching components. Use a Service Locator. (Multiweek) Use Nystrom, apply assertions
Home	View Lynda.com : Foundations of Programming: Object-Oriented Design Study Nystrom : http://gameprogrammingpatterns.com/contents.html Classics : 1, 2 Command, 4 Observer, 6 Singleton, 7 State. Recap GamPats : 9 The Game Loop, 10 Update Method, 14 Component Apply ansertions

Week 5:

Lecture	Design based: UML to go, Overview and Use Case, Class, Sequence, State
Lab	Do your diagrams for Architectural, a Class Diagram, and Sequence diagrams, purpose,
Home	Lynda.com : Foundations of Programming: Object-Oriented Design Lynda.com : Foundations of Programming: Test-Driven Development. Nystrom reread and continue.

Week 6:

Lecture	Patterns : Event Queue, Command, State, Singleton
Lab	Decouple : Add the Event Queue for low and high level Events. Deliver CollisionEvents to the involved objects resulting in higher order events handled by their observers. Also have input be handled using the Command Pattern. Maintain your diagrams
Home	Nystrom 15 Event queue.

Week 7:

Lecture	Code Quality: Assertions and Unit Testing
Lab	Extend you game with variations in paddle, ball and booster look and behavior using State Pattern. Maintain your diagrams.

Home	Continue work on the assignment. Use lab to validate state.

Week 8:

Lecture	Questions, Todo’s and Optionals Double Buffer, Object Pool, Prototype, Spatial Partitioning, Test Exam
Lab	Validation of delivered labwork
Home	Example exam and Theory study Complete the assignment, and upload. This week is deadline !!!!

Resources

Written exam refererences (questions in exam are taken from these resources)

- Overall
  - Lynda.com :
    - Foundations of Programming: Object-Oriented Design
    - Foundations of Programming: Test-Driven Development
- Design Patterns
  - http://gameprogrammingpatterns.com/contents.html
  - Chapters : 1, 2, 4, 6, 7, 9, 10, 14, 15, 16

Extra (non exam)

- Linda : Learning S.O.L.I.D. Programming Principles
- UML :
  - http://www.sparxsystems.com/resources/uml2_tutorial/
  - http://www.ibm.com/developerworks/rational/library/769.html
  - http://www.ibm.com/developerworks/rational/library/3101.html
- Unity Patterns :
  - https://www.safaribooksonline.com/library/view/head-first-design/0596007124/
  - http://www.rivellomultimediaconsulting.com/unity3d-mvcs-architectures-strangeioc/
  - http://www.rivellomultimediaconsulting.com/unity-csharp-state-management/
  - http://www.rivellomultimediaconsulting.com/unity3d-architectures-for-games/
- Threading
  - Albahari, Threading in C#. http://www.albahari.info/threading/threading.pdf
- Tools
  - UML : http://astah.net/editions/community preferred.
  - UML : https://www.draw.io/ !!! carefull no meaning to symbols
  - Nunit : http://www.nunit.org/

3 Exam Criteria and Judgement

Procedure
- The exam is based on the mentiones obligational resources and has both multiple choice and open questions
- Assignment is delivered through blackboard, Week 8, Friday, 23:59 hours.
- Written exam and assignment both weight 0.5 for the final grade.

Grading form assignment

Be aware that good includes sufficient and excellent includes good !
The Assignment is a design document and a demo evolved from the simple GAGame engine.

	Criteria	Insufficient	Sufficient	Good	Excellent
1	Design pattern	Poorly used patterns	Proper use of required patterns. Clean code.	Applying extra patterns.	Discussing use of patterns referring to external resources.
2	Designing Diagrams : UML	No or poorly used UML	Documented design using required UML. Use Case, Class, Sequence, State	Extensive use of explicit types of associations.	Additional UML diagrams used.
3	Code Quality	No enforced Code Quality	Assertions used properly. Clean code.	+ Logged peer reviews	+ Unit Tests

Thats all folks.