Component baseret systemer

***Student Name:* Mads Wrang Sigsgaard**

***Student Exam number:* 4118544**

***GitHub User:* DenFlyvendeGris**

***GitHub Repo:*** [**https://github.com/DenFlyvendeGris/AsteroidsFX**](https://github.com/DenFlyvendeGris/AsteroidsFX)

**Abstract**

Describe the problem that the report addresses in context of the game domain.

Outline how the developed game addresses the requirement – its key characteristics and fundamental principles (establishing a solution).

**Introduction**

The asteroid game is a well know game that was created back in 1979 and had a great impact on the software industry as it had simple yet challenging gameplay. This arcade game has since been recreated a lot of times with different spins on the gameplay and visuals. Apart from gameplay and visuals, it has also been created with different software structures. The simplicity of the game makes it a perfect practice to try different software structures like component-based software engineering (CBSE), which is what I have tried with the asteroids game. To create a CBSE it is essential to create a project that is modular and easily changeable, in other words it’s important to have it be closed to modification and open to extension.

The introduction must describe the game.

**Requirements**

|  |  |  |
| --- | --- | --- |
| Functional Requirements | | |
| ID | Title | Description |
| 1 | Player Module | The function of a player module is to draw the player on the world map, and give the player logic for movement |
| 2 | Enemy Module | The function of the enemy module is to spawn multiple enemies throughout the game runtime, and give them movement like the player, but they need some sort of movement pattern, and that is created by randomizing their movement. |
| 3 | Asteroid Module | The asteroid module is there to create asteroids and give them a state which is Large, Medium, or small. This is used in collision module to either split an asteroid or delete it. |
| 4 | Bullet Module | The bullet module is creating a bullet and removes them when they are outside of the height and width of the screen. Both the player and enemy use the bullet module |
| 5 | Common Module | The common module starts the game and draws entities on the world from the startup point, it is also the module that has all the entities and has the logic for deletion from the world. |
| 6 | Collision Module | The collision module is responsible of the collision logic. The entity class has onCollision logic that is used in Player, Enemy and Asteroid to respond in different ways depending on what entities collide with each other |

|  |  |  |
| --- | --- | --- |
| Non-Functional Requirements | | |
| ID | Title | Description |
| 1 |  |  |
|  |  |  |

Describe the component-based game in terms **of interface contracts**, functional and non-functional requirements.   
The game must include **Player**, **Enemy**, Asteroids, **Weapon** and Map components.

The Player, Enemy and Weapon components must implement service provided interfaces that allow the components to be updated and removed without recompilation.

**Analysis**

Analysis describes only **what** the system should do and not **how** it is done.

In analysis, you can come up with a rough draft of the interfaces and the entities of the game.

Furthermore, you should document use cases/gameplay, the object model using a UML class diagram and the communication between components with sequence diagrams.

**Design**

The design describes **what** the structure of the system should be to fulfill the requirements.

Document the architecture and abstractions of the system.

Design develops those abstractions into realizable components.

Describe and sketch the **component models** of the game using a UML component diagram, see [[UML]](https://mcas-proxyweb.mcas.ms/certificate-checker?login=false&originalUrl=https%3A%2F%2Fyoutu.be.mcas.ms%2FKQUGFFN4M90%3FMcasTsid%3D15600&McasCSRF=06d27f0d1db0da31f3bf2a820cad1cc27f6a89a4b27591733e8f071c9ca03510) .

The component contracts in the system must be described in terms of pre- and post-conditions.

Furthermore, the different elements of the game and how they are connected must be described.

**Implementation**

In implementation, you document the implementation (code) of the components from design.

Describe the details of how the components are registered and accessed.

How are reliable dependencies and strong encapsulation enforced in your project?

What component models are applied and where in the source code?

Provide a descriptive explanation of each element in the implementation and provide arguments for your choices.

You should describe how you register components and how you access them.

Note, you are allowed to reuse the game logic, but you must implement the **[GameLab], [JavaLab]**, **[JPMSLabs]**, **[SpringLab],** **[TestLab]** and [MicroServiceLab] labs based on your own GitHub branches.

**Test**

Describe how experimental validation was performed through deployment of the game on top of the component container in a real setting.

Test the system's software-abilities such as dynamic updates using integration and unit test.

**Discussion**

Discuss how well the game solved the identified essential problems (module updates etc.).

To which extent did your design meet the requirements?

**Conclusion**

First summarize the report.

Remember that you are summarizing the report for a reader that has read the introduction and the body of the report already and has a strong sense of key concepts and applied technologies.

Explain the potential impacts of your system in relation to the main issue.

Direct future work directions related to the main issue.

However, this should not be seen as an opportunity to develop new ideas in significant detail and should be clearly linked to the work described in your report.

**References**

[Please use IEEE citation style](https://mcas-proxyweb.mcas.ms/certificate-checker?login=false&originalUrl=https%3A%2F%2Fieeeauthorcenter.ieee.org.mcas.ms%2Fwp-content%2Fuploads%2FIEEE-Reference-Guide.pdf%3FMcasTsid%3D15600&McasCSRF=06d27f0d1db0da31f3bf2a820cad1cc27f6a89a4b27591733e8f071c9ca03510),