Software Engineering

LECTURE 4 (2): Group Project Overview

Key Activities

- ❑ Teaming and Project selection
- ❑ Forming subgroups and assigning mini-projects
- ❑ Build Requirements Analysis Documents (RAD)
- ❑ System Design Documents (SDD)
- ❑ Working Software and Documentations

Problem Statement

- Short description about the project chosen from which you can derive functional and nonfunctional requirements.
  - Purpose
  - Scope
  - Objectives and Success criteria
- Choose a project from:
  - https://nevonprojects.com/year-projects-for-computer-engineering/ or
  - https://www.upgrad.com/blog/software-development-project-ideas-topics-for-beginners/

Form Sub-groups

- Each project group has 6±1 students
- Split into 3 sub-groups of 2 students (ideal) or sub-groups with 1 or 3 students
- Each sub-group designs their own "mini-project", which must be part of the overall project developed by this group
  - "Requirements projection" through the development lifecycle, from requirements, through design, to code

Divide Work to Sub-groups

1. Pretend that your subgroup of one or two (or three) is going to develop a smaller version of the system-to-be ("sub-system"), with only one or two functional features ← **Mini-projects**
2. Identify the external entities that interact with your sub-system ("actors") and define what kind of problem needs to be solved
3. Describe in detail how your sub-system will work when seen from the user's viewpoint.
   — Others should see your sub-system as a "black box"
4. See what interactions can be identified between different sub-systems.
5. Other sub-systems within your project that interact with your sub-system should be treated as "third-party" systems, external to your sub-system.

Other Considerations

- The goal is not to develop a software blindly.
  - Learn how to successfully engineer a software system in a systematic and disciplined way!
- Teaming
  - If you have difficulty joining a proper group, you can form a mini-team of 2 or 4 students to solve partial problems
- Which method to follow is your choice!
  - Waterfall, Iterative, or Agile
- All the artifacts and outputs would be managed by Git/GitHub

The 1st Checkpoint

- Due by Apr. 25th

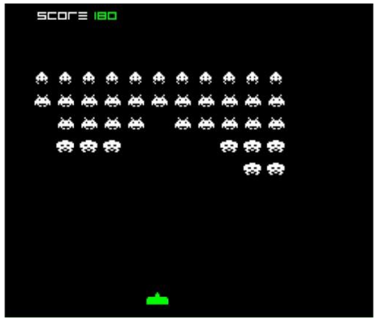
- Problem Statement
- Gathered Requirements
  - In IEEE-830 style or User stories
  - Functional and Nonfunctional
- System Model
  - Use Case model (w/ Traceability Matrix)
  - Domain model (E-C-B) (w/ Traceability Matrix)
  - User Interface mockups (if any)

Problem Statement: Space Invaders

❑ The Problem

The market for computer games is moving towards complex, photo-realistic 3D games. However, such games have long startup times and high learning curves. They also require a lot of power and time.

Solitaire, Snake or Angry Birds are examples for the success of simple and easy to use games. They have a high fun factor, no learning curve and short startup times. As the duration of these games is short, they can be played in nearly every situation. We want to have a simple and easy Space Invaders game and offer it for you for boring lectures. See the exemplary user interface below to get a feeling how the app should look like/



Problem Statement: Space Invaders

Scenarios

At the start of the SI app, the player (she) has the chance to type in their player name. After typing in her name, the player can see a menu, which offers 3 options: “Start”, “High scores” and “Quit”. She can navigate through these 3 options with the key arrows on her keyboard and can select one option by clicking the space key.

After selecting “Start”, the player selects the 4th level and can start playing the game. She steers the space ship to the left and the right and shoots rockets in order to destroy invaders. In case a rocket hits an invader, it disappears and a crash sound plays. The goal of the game is to remove all invaders from the game board as fast as possible.

After the player has destroyed all invaders, the game displays the total time that she needed. She is able to submit her personal high score in order to compare with other players of the SI application.
While the game is running, there is music, when pausing the game, the music stops. During the game, the player can inspect the already elapsed time and see the status how many invaders she already destroyed.

Requirements (IEEE-830)

REQ-1 (FR / 4): The system should show the menu, which offer options (“Start”, “High scores”, “Quit”) after players type their name.
REQ-2 (FR / 4): The system should allow players to navigate options of menu using the arrow keys on their keyboard.
REQ-3 (FR / 4): The system should allow players to select options of menu using the space keys on their keyboard.
REQ-4 (FR / 3): The system shall show the menu, which select level of game (1~4 level) after selecting “Start”, and then allow player can select level of game using their keyboard and start playing game.
REQ-5 (FR / 1): The system shall allow players to control spaceship with using their keyboard
REQ-6 (FR / 1): The system shall provide a rocket that can be shoot from spaceship in order to destroy invaders.
REQ-7 (FR / 3): The system should play crash sound when rocket hits an invader.
REQ-8 (FR / 4): The system should display total time what player needed after the player has destroyed all invaders.
REQ-9 (FR / 5): The system should submit player personal score in order to compare with another player.
REQ-10 (FR / 3): The system should pause music when pausing the game and play music when running game.
REQ-11 (FR / 1): The system should operate in any environment with keyboards.
REQ-12 (NFR / 4): The system should have a space where players can write their name before starting game.
REQ-13 (NFR / 2): The system shall allow the player can see the already elapsed time during the game.
REQ-14 (NFR / 2): The system shall allow the player can see the status how many invaders player already destroyed during the game.
REQ-15 (NFR / 1): The system should set the goal of the game (remove all invaders from game board as fast as possible)

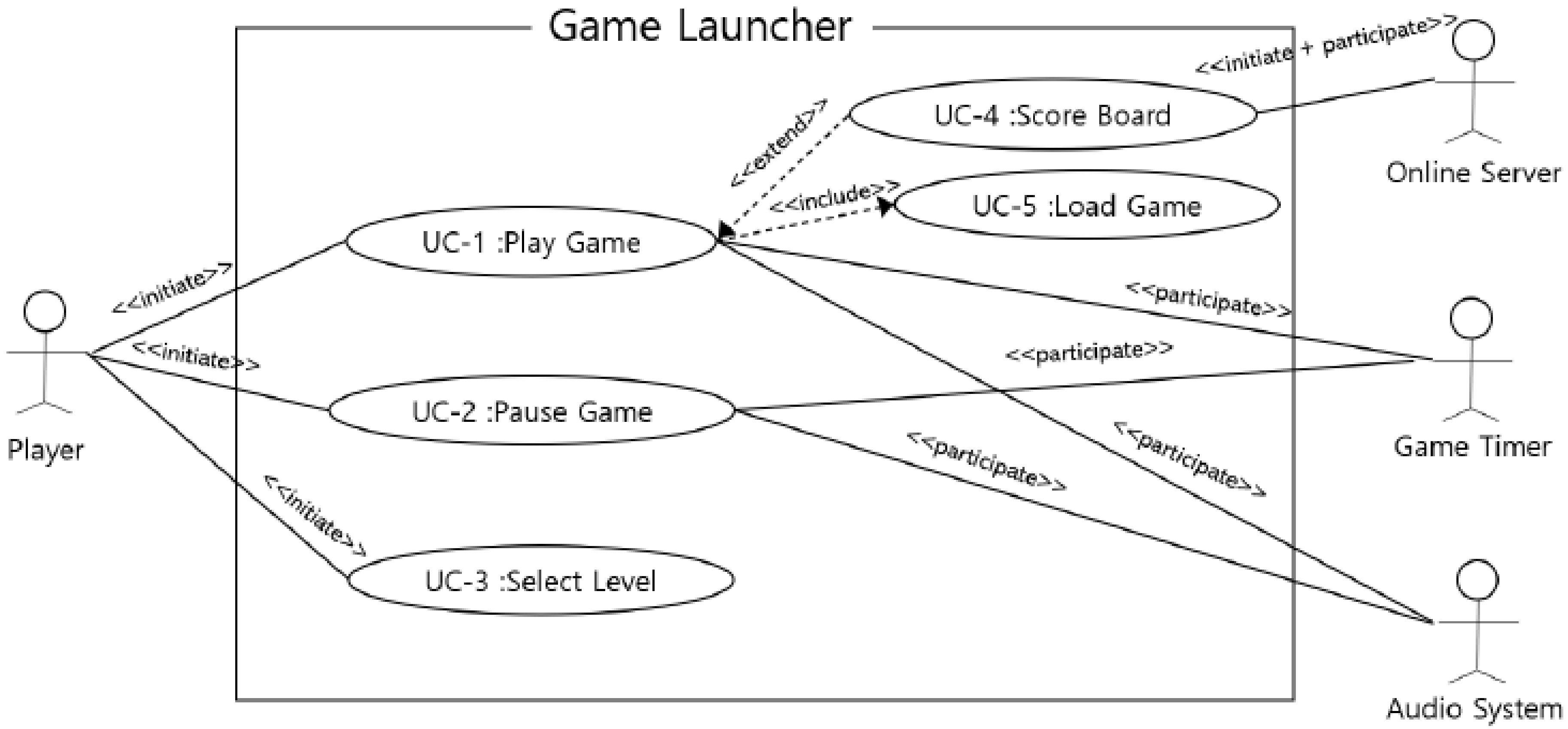
Requirements (User Stories)

FR1: As a game plyer, I will be able to type in my name when the SI app starts.
FR2: As a game plyer, I will be able to navigate 3 options: “Start”, “High scores” and “Quit” by using the key arrows on my keyboard and select one option by clicking the space key.
FR3: As a game plyer, I will be able to choose 4th levels after selecting “Start”.
FR4: As a game plyer, I will be able to steer the spaceship to the left and the right and shoots rockets
FR5: The invaders will be disappeared with a crash sound when hit by a rocket.
FR6: As a game plyer, I will be able to pause the game.
FR7: As a game plyer, I will be able to get the score and total time when all invaders are disappeared.
FR8: As a game plyer, I will be able to submit my personal high score in order to compare with other players of the SI application.
FR9: As a game plyer, I will be able to listen a music while the game is running but not in pausing situation.
FR10: As a game plyer, I will be able to inspect the already elapsed time and see the status how many invaders I already destroyed.
NFR1: This game will be able to have a high fun factor, no learning curve and short startup times.
NFR2: This game will be able to be played in nearly every situation.
NFR3: The game will be able to have simple UI which has a black background, white invaders, and a green spaceship like “Gallag”.

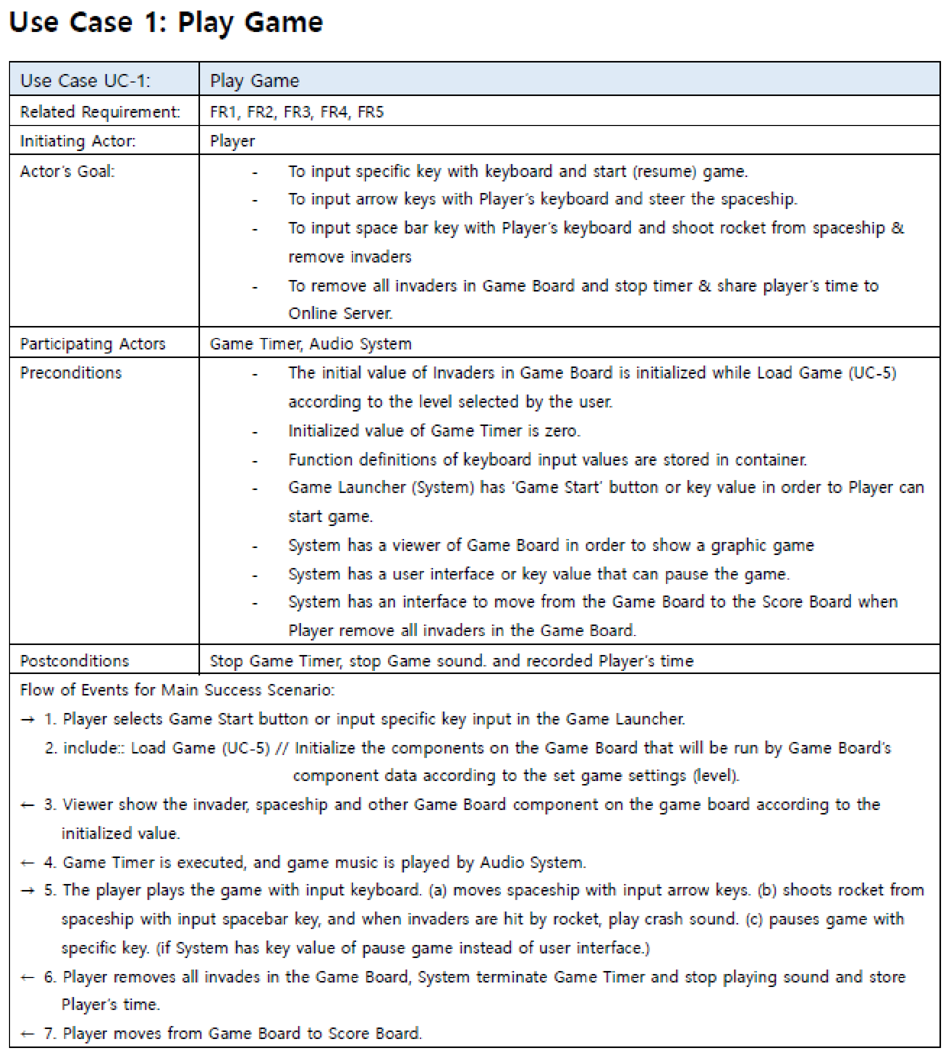
Use Cases

Actor	Actor's Goal	Use Case Name
Player	To input specific key with keyboard and start (resume) game.	UC-1: Play Game
Player	To input specific key with keyboard and pause game.	UC-2: Pause Game
Player	To select level of game, and change type of invader and arrangement of invader in Game board.	UC-1, UC-3: select level
Player	To input arrow keys with Player's keyboard and steer the spaceship.	UC-1
Player	To input space bar key with Player's keyboard and shoot rocket from spaceship & remove invaders.	UC-1
Player	To remove all invaders in Game board and stop timer & share Player's time record to Online Server.	UC-1, UC-4: Score Board
Game Timer	To record Player's time when game running.	UC-1, UC-2
Online Server	To upload Player's time data and download other's time data& build Score Board.	UC-4
Audio System	To control the audio of game.	UC-1, UC-2

Use Cases


Game Launcher
UC-1 :Play Game
UC-2 :Pause Game
UC-3 :Select Level
UC-4 :Score Board
UC-5 :Load Game
Player
Online Server
Game Timer
Audio System
<<initiate>>
<<initiate>>
<<initiate>>
<<extend>>
<<include>>
<<initiate + participate>>
<<participate>>
<<participate>>
<<participate>>
<<participate>>

Use Cases



Use Case 1: Play Game

Use Case UC-1:	Play Game
Related Requirement:	FR1, FR2, FR3, FR4, FR5
Initiating Actor:	Player
Actor's Goal:	- To input specific key with keyboard and start (resume) game. - To input arrow keys with Player's keyboard and steer the spaceship. - To input space bar key with Player's keyboard and shoot rocket from spaceship & remove invaders - To remove all invaders in Game Board and stop timer & share player's time to Online Server.
Participating Actors	Game Timer, Audio System
Preconditions	- The initial value of Invaders in Game Board is initialized while Load Game (UC-5) according to the level selected by the user. - Initialized value of Game Timer is zero. - Function definitions of keyboard input values are stored in container. - Game Launcher (System) has 'Game Start' button or key value in order to Player can start game. - System has a viewer of Game Board in order to show a graphic game - System has a user interface or key value that can pause the game. - System has an interface to move from the Game Board to the Score Board when Player remove all invaders in the Game Board.
Postconditions	Stop Game Timer, stop Game sound. and recorded Player's time
Flow of Events for Main Success Scenario: → 1. Player selects Game Start button or input specific key input in the Game Launcher. 2. include:: Load Game (UC-5) // Initialize the components on the Game Board that will be run by Game Board's component data according to the set game settings (level). ← 3. Viewer show the invader, spaceship and other Game Board component on the game board according to the initialized value. ← 4. Game Timer is executed, and game music is played by Audio System. → 5. The player plays the game with input keyboard. (a) moves spaceship with input arrow keys. (b) shoots rocket from spaceship with input spacebar key, and when invaders are hit by rocket, play crash sound. (c) pauses game with specific key. (if System has key value of pause game instead of user interface.) ← 6. Player removes all invades in the Game Board, System terminate Game Timer and stop playing sound and store Player's time. ← 7. Player moves from Game Board to Score Board.	

Domain Model for UC-1

Extracting the responsibility

Responsibility Description	Type	Concept Name
Coordinate actions of all concepts associated with a use case, a logical grouping of use cases, or the entire system and delegate the work to other concepts.	D	Controller
Defining functions about keyboard input.	K	CheckKey
Value of Player keyboard input	K	KeyboardEntry
Container for game component's data of running game (spaceship, invaders, rocket)	K	CurrentData
Operate the Sound System to play sound / stop playing sound	D	AudioOperator
Operate the Game Timer to record time / stop record time	D	TimerOperator
Container for store Player's time data follow game level.	K	TimeData
Display game component about current game state.	D	Viewer
Calculate changed value of game components due to Player input.	D	Controller

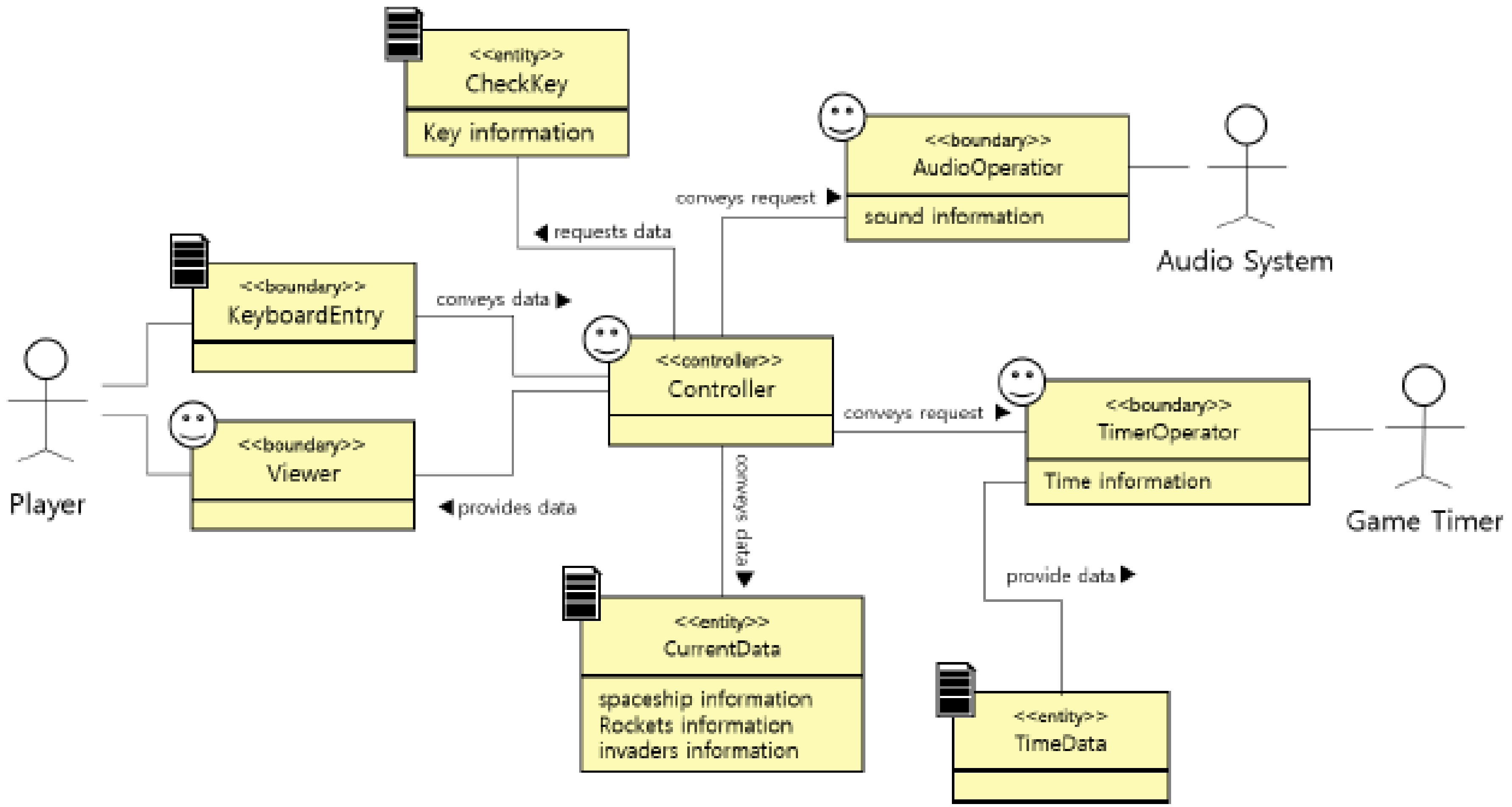
Extracting the Associations

Concept pair	Association description	Association name
Controller <-> CheckKey	Controller asks what operation definition of the keyboard input values the user entered.	requests data
KeyboardEntry <-> Controller	KeyboardEntry value is passed to Controller	conveys data
Controller <-> CurrentData	Controller passes data that calculates game component (spaceship, invaders, rocket) change according to the Player input with keyboard.	conveys data
Controller <-> AudioOperator	Controller passes sound operation requests to AudioOperator.	conveys requests
Controller <-> TimerOperator	Controller passes Timer operation requests to TimerOperator.	conveys requests
TimerOperator <-> TimeData	TimerOperator provides data of recorded Player's time to TimeData.	provides data
Controller <-> Viewer	Controller provide game component for display to Viewer.	provides data

Extracting the Attributes

Concept	Attributes	Attribute Description
CheckKey	key information	Key information of Player input keys with Keyboard and define the corresponding action.
CurrentData	spaceship information	Current data of spaceship.
	rockets information	Current data of rockets.
	invaders information	Current data of invaders.
AudioOperator	sound information	Information of sound to play.
TimeOperator	time information	Information of Player's time.

Domain Model for UC-1



<<entity>>
CheckKey
Key information
<<boundary>>
AudioOperatior
sound information
Audio System
conveys request
requests data
<<boundary>>
KeyboardEntry
conveys data
<<controller>>
Controller
Player
<<boundary>>
Viewer
provides data
conveys request
<<boundary>>
TimerOperator
Time information
Game Timer
conveys data
<<entity>>
CurrentData
spaceship information
Rockets information
invaders information
provide data
<<entity>>
TimeData