Team Project

INF1009 Object Oriented Programming

Objective

- Put the object-oriented principles that are taught in class to use in a real application.

- OOP helps the design
  - Objects in the program $=$ objects in the user's world.
  - How these group of objects will work together?

Learning Outcomes

- Create objects of different classes in the same application
- Allow objects to communicate with each other
- Create more complex objects by combining simpler ones
- Derive new classes from existing ones
- Extend definition of existing classes
- Overriding methods

Keep in Mind!!

- Programming is all about solving problem
- We are the client proposing a problem, you must solve the problem
- Code should be well designed and implemented following core principles for programming and software development

What do you have to do?

Build a Simulation

Simulation

Abstract Engine

Week 1 – 7

Logic Engine

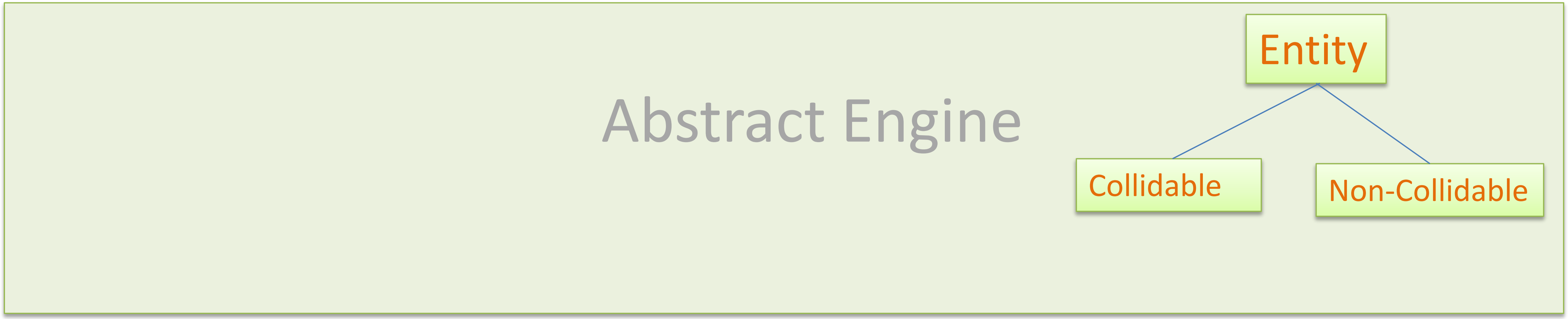
Week 8 – 12

Simulation

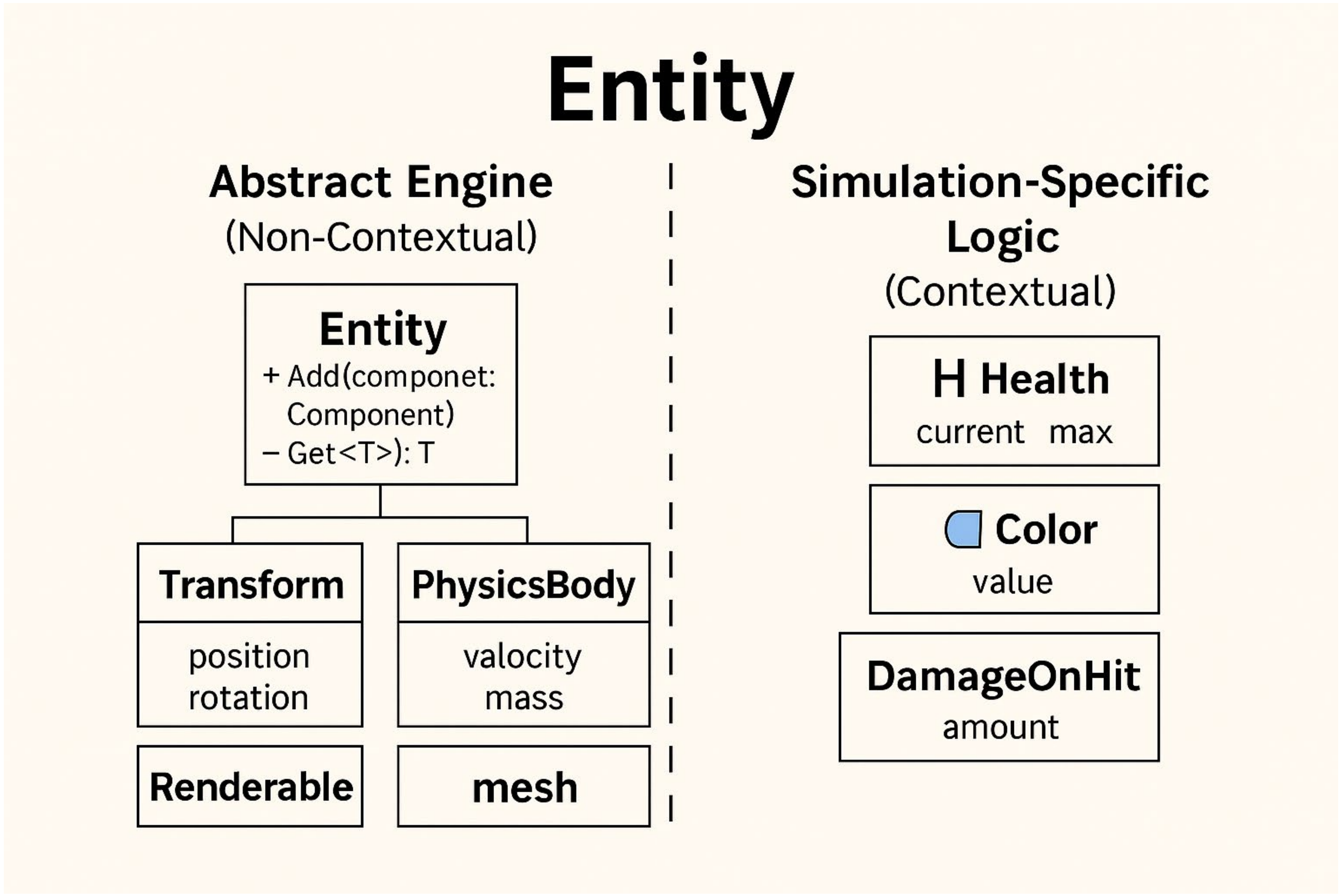
Abstract Engine

- ***The Abstract Engine is NOT the actual simulation or game.***
- ***It is a generic foundation that provides core functionality.***

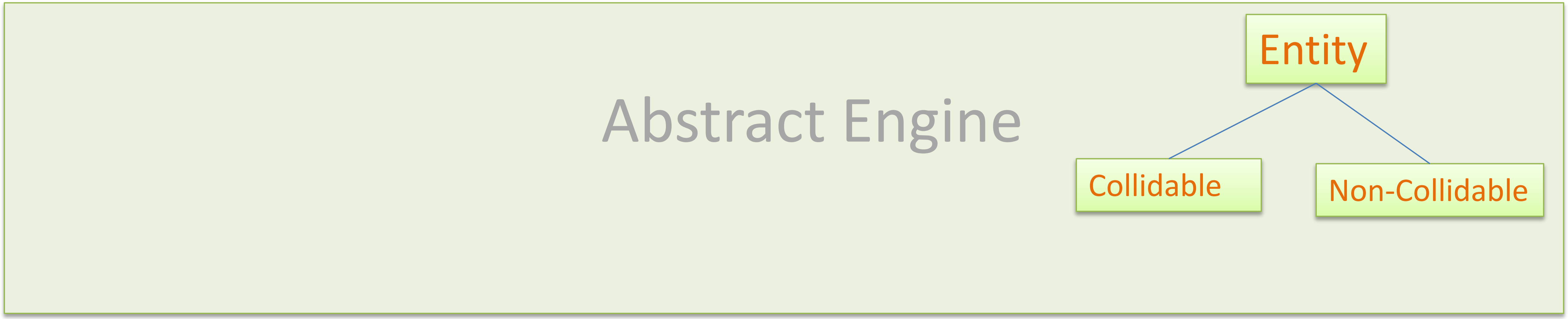
Simulation



- ***The Abstract Engine is NOT the actual simulation or game.***
- ***It is a generic foundation that provides core functionality.***
- ***It exists outside any specific context (not tied to rockets, cars, or traffic lights).***
- ***Its purpose is to offer reusable components that can be applied to build different simulations or games.***

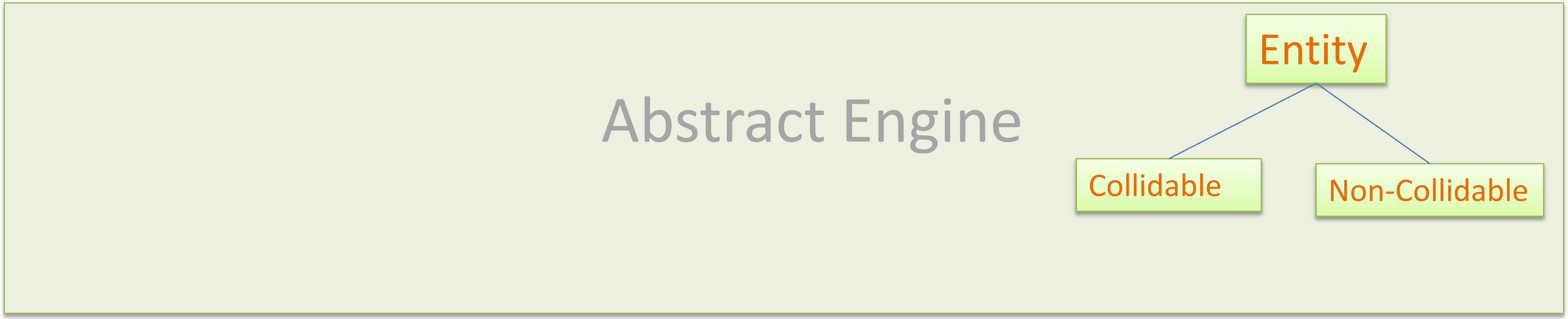

Entity
Abstract Engine
(Non-Contextual)
Entity
+ Add(componet: Component)
− Get<T>): T
Transform
position
rotation
PhysicsBody
valocity
mass
Renderable
mesh
Simulation-Specific Logic
(Contextual)
H Health
current max
Color
value
DamageOnHit
amount

Simulation



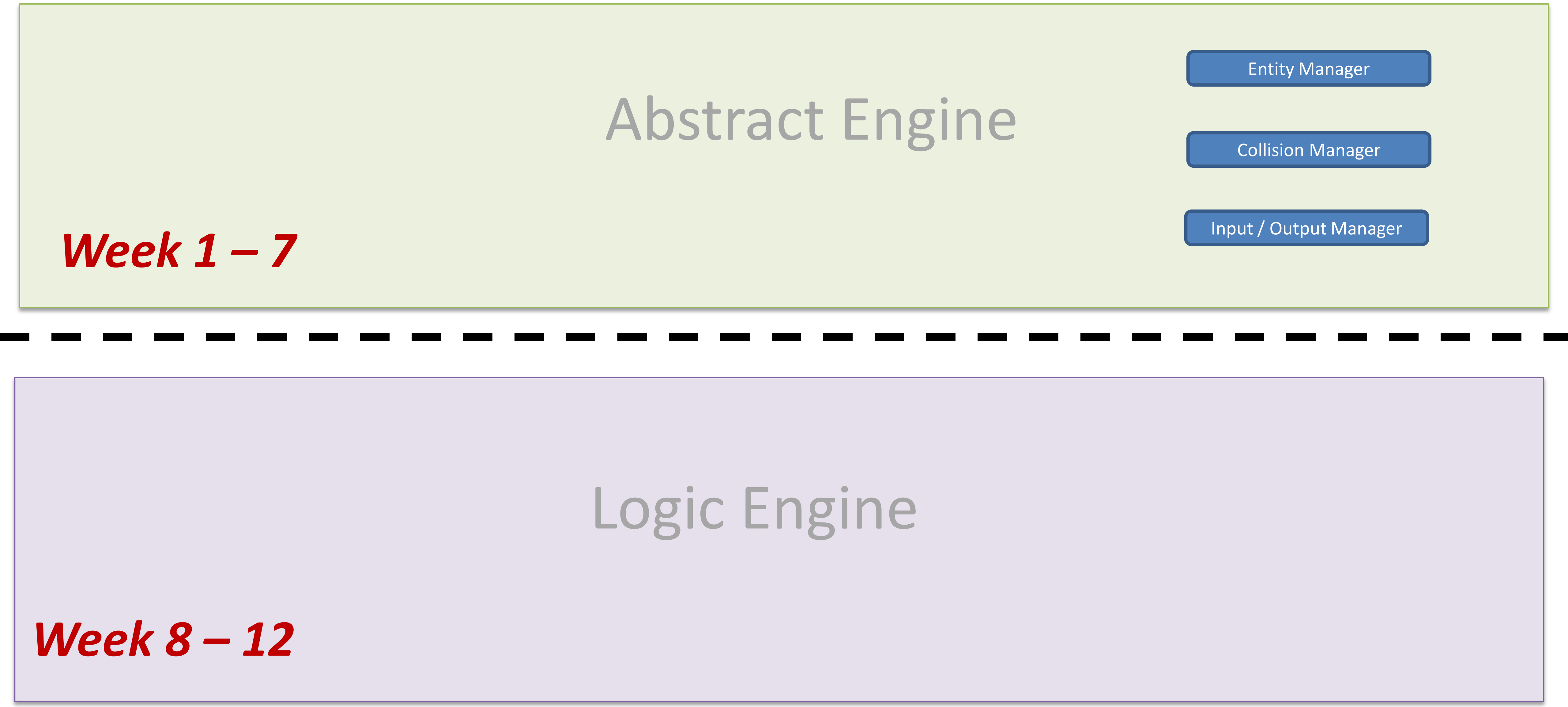
- ***The Abstract Engine is NOT the actual simulation or game.***
- ***It is a generic foundation that provides core functionality.***
- ***It exists outside any specific context (not tied to rockets, cars, or traffic lights).***
- ***Its purpose is to offer reusable components that can be applied to build different simulations or games.***
- ***Think of it as the toolbox or framework that your Logic Engine will use.***

Simulation

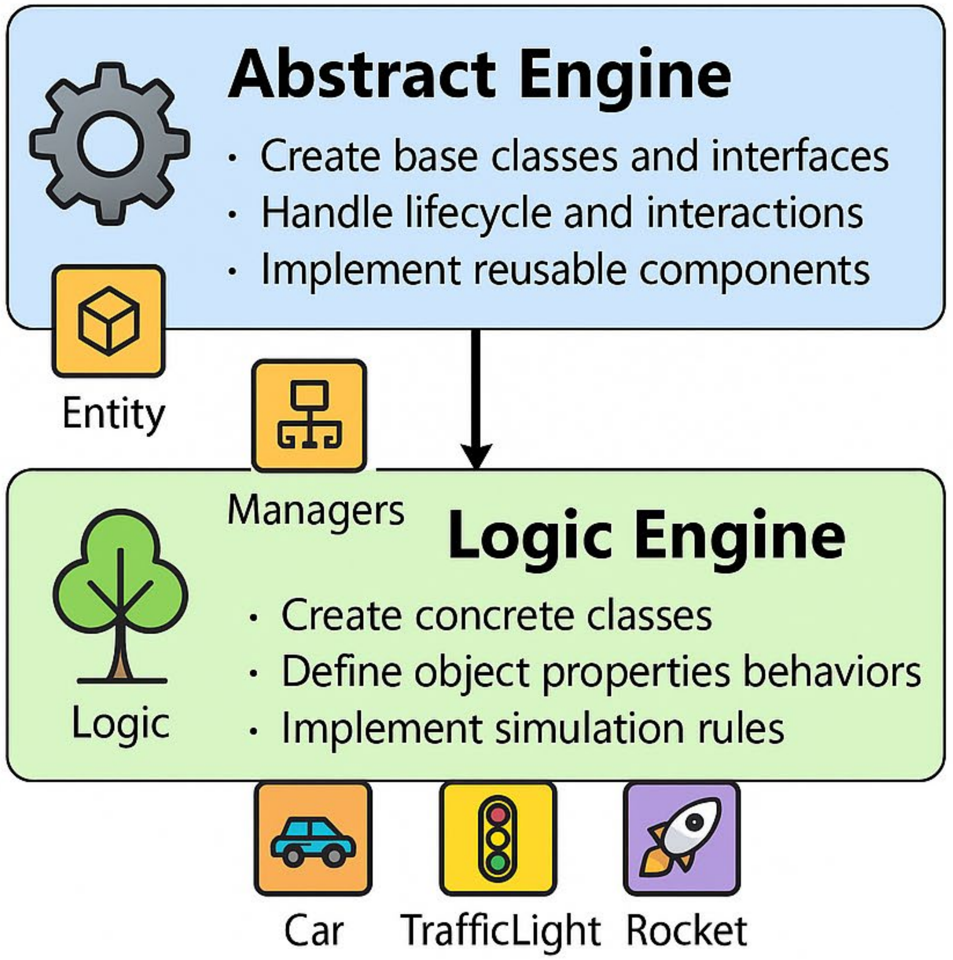


- ***Delete the logic layer, keep the engine – boom, you're ready to build something totally new without starting from scratch.***
- ***One engine, endless possibilities – traffic sim today, rocket sim tomorrow, maybe even a game next week.***
- ***Think of it as your coding superpower – build once, reuse forever.***

Simulation


Abstract Engine
Entity Manager
Collision Manager
Input / Output Manager
Week 1 – 7
Logic Engine
Week 8 – 12

Simulation



Abstract Engine

- Create base classes and interfaces
- Handle lifecycle and interactions
- Implement reusable components

Logic Engine

- Create concrete classes
- Define object properties behaviors
- Implement simulation rules

Develop a Simulation

- There are many **important components** in a simulation
  - entities, environment, logic etc.
- From the developer's perspective,
  - what are the **main objects** in the simulation ?
  - What are these objects ?
  - What are the main characteristics of these objects?
- How can your team **employ the concepts of Object Oriented programming** to develop a simulation

Project Description

- Develop a simulation environment
  - You can only use **libgdx**
- The project has to be developed in two parts:
  1. The management of simulation elements is all part of the **abstract engine** (non-contextual) : Part 1
  2. **Simulation specific logic** (contextual): Part 2

- Most important to focus on a good engine with working components (no shortcuts to develop the components, e.g. no libraries).

Good Programming Practices: SOLID

- The **S**ingle Responsibility Principle
- The **O**pen-Closed Principle
- The **L**iskov Substitution Principle
- The **I**nterface Segregation Principle
- The **D**ependency Inversion Principle

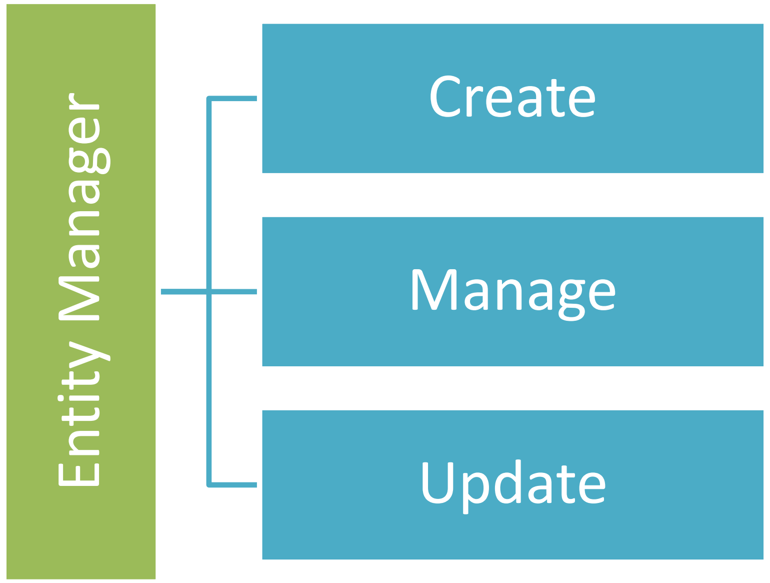
Abstract Engine : Must Have

- Scene Management
- Entity Management
- Collision Management
- Movement Management
- Input/Output Management

Project details: Scene management

- Load and unload different scenes
- Transitioning between different scenes

Project details: Entity management



Creating, Managing and Updating all entities within the simulated world

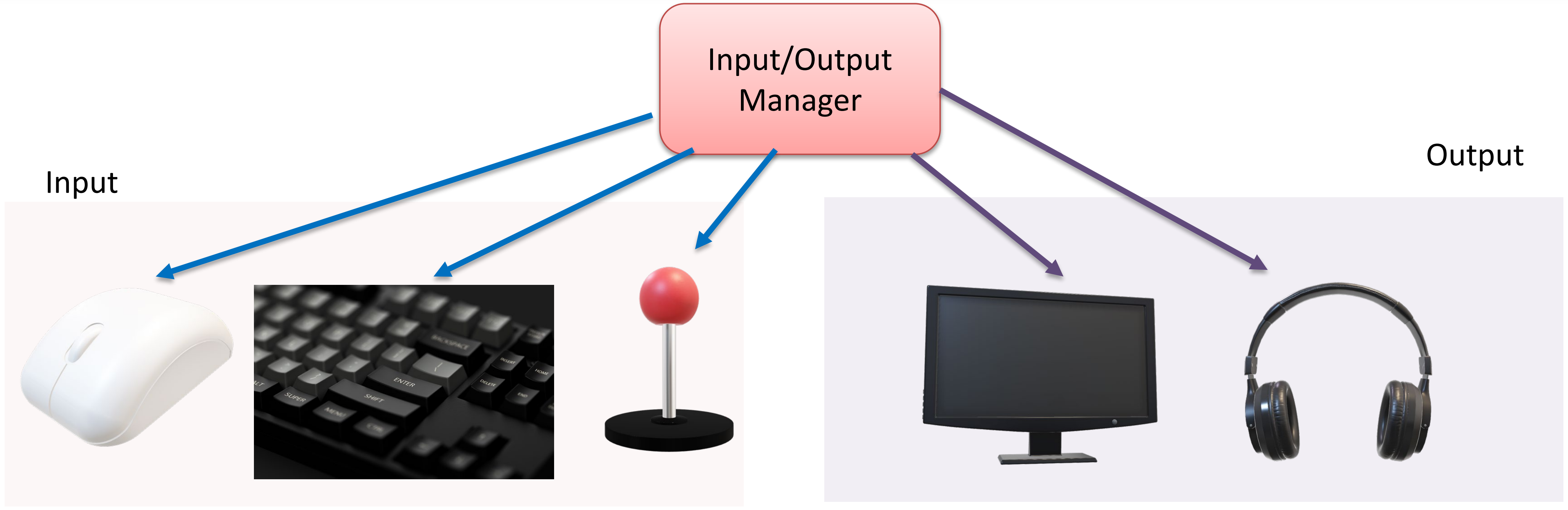
Project Details: Collision Management

Handling interactions between different objects or entities within the simulated world

$$\begin{tikzcd} & \text{Collision Management} \arrow[dl] \arrow[dr] & \\ \text{Collision detection} & & \text{Collision resolution} \end{tikzcd}$$

Project Details: Input/Output management

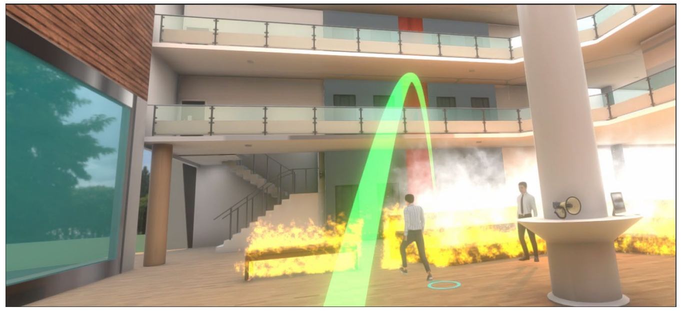
Managing all user input and output



Project Details: Movement management

- Movement of different non-playing entities

- What properties will control the movement?



What do you have to build

- You must solve the problem using code to build a simulation:

- The problem must be solved well, so the client is happy, target audience would want to play/run it

What needs to be showcased (Part 1)

- Need to showcase a prototype to demonstrate the abstract engine technology and game mechanics
- Proper usage of OOP concepts
  - Classes
  - Objects
  - Inheritance
  - Polymorphism
  - Error Handling
- Re-usablilty of code
  - Modularity
- Functionalities
  - Various features of the game

How to code

- UML diagram
  - Identify the main entities to create classes
  - Think how you want to structure and connect these classes
- Code
  - Store code in a repository
  - Collaborate with your team-mates
  - Use it for the entire project
- Code structure
  - File, class and function naming convention should be followed (they should be self explanatory)

Project Deliverables: Part 1
Milestone 1 (Hurdle)

UML diagram

- Deadline: **Week 4**
- UML diagram for the abstract engine
- Class hierarchies
- Relationships between different classes and interfaces

What needs to be submitted

- Report
  - Professionally written with clear logic and structure
- Code
  - Dropbox submission of the entire code of prototype
- Demo
  - Simple demo of the simulation engine

  E.g. what if we have 400 drops or we want to extend to 4 players how can it be done in the engine

- Presentation
  - Presentation Slides to showcase exactly what the team has done
  - One short video to demonstrate all features and design of the system
  - Innovation and functionalities

Project Deliverables: Part 1
Final Submission Week 7

Report

Code

Presentation

Video

Peer Evaluation

Remember

- Aim for Innovation/Complexity/Solving for the Problem

- Should focus on solving a problem better than other solutions to same problem (market value)

- Think whether proposed solution will work for client/target
audience