Junior Programmer: Resource management simulation

Project brief

Application concept

Overview

In this project, you will implement functionality for a **resource management simulation application**. Resource management is at the heart of many genres of games, but simulations are also commonly used in educational contexts (for example, to explore sustainability and economics) and in industry.

This type of project is a particularly good fit to help you explore programming systems and architecture, as they often include:

- User interaction, to enable them to influence the simulation
- Transitions between scenes, to enable further customization by the user
- Systems designed for extension, to increase the complexity of the simulation

Reference examples

Some examples of resource management applications that you might want to explore include:

- **Fishbanks:** A multiplayer renewable resource management simulation for educational use
- **Age of Empires** or **Civilization**: Classic strategy games requiring resource management

---

Junior Programmer: Resource management simulation

Your task checklist

Before you go through the brief in detail, here's a high level checklist of what you will do in this project:

Scene management

- ⛔ Create transitions between two scenes
- ⛔ Configure buttons so the user can control those transitions
- ⛔ Configure a button to exit the application (or exit Play mode, in Unity Editor)

Data persistence

- ⛔ Configure buttons in one scene to apply a chosen color to objects in a second scene
- ⛔ Save the last color chosen by the user and preselect it the next time the application is launched

Inheritance and polymorphism

- ⛔ Create a new type of object in the simulation, with a variation on behavior derived from a base class

Abstraction

- ⛔ Refactor to reduce duplicate code and improve reusability

Encapsulation

- ⛔ Use getters and setters to protect data from misuse

Optimizing code

- ⛔ Profile example code to identify basic optimization issues

---

Junior Programmer: Resource management simulation

Unity project overview

Important: When you first open the project, the warehouse simulation will have basic functionality but the application will not work!

The application

Scene in the project

There are two scenes for the application in the Unity project:

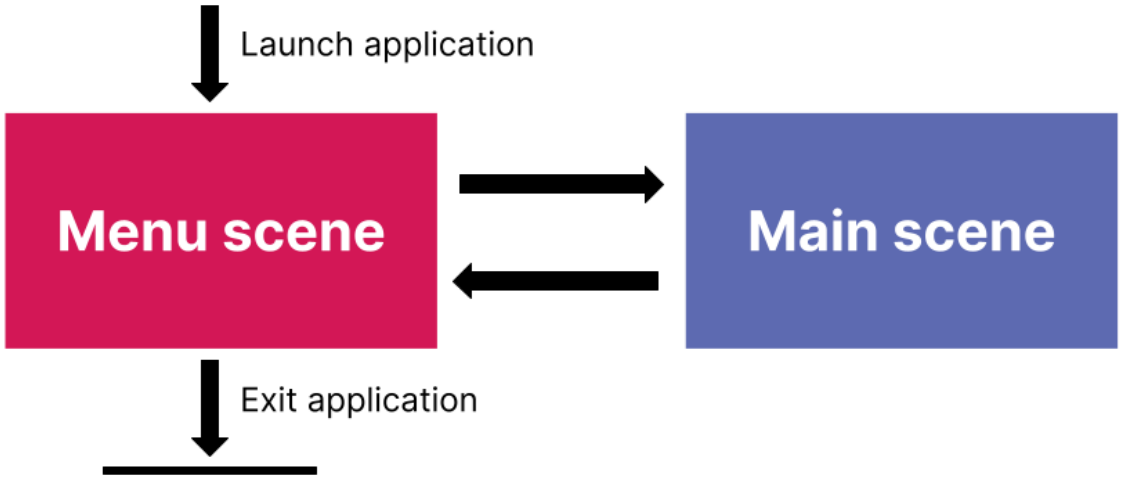
1. A starting menu (Menu), where users can launch the simulation and close the application
2. A simulation scene (Main), which is styled as a warehouse

Note: There is also an additional scene called Optimization — this is an optimization case study that you'll use to profile code and identify issues.

User interactions

You will implement user-controlled transitions:

- Between the two scenes
- To exit the app, from the Menu scene only



User interactions

The user needs to be able to:

- Launch the simulation scene from the starting menu
- Return to the starting menu from the simulation scene
- Exit the app (or exit Play mode, for in-Editor testing)
- Select a color to apply to the Transporter Units (forklifts) in the simulation

The last color chosen by the user should also be pre-selected the next time that they launch the application.

Junior Programmer: Resource management simulation

The warehouse simulation

The basic simulation

We've created a basic simulation (in the Main scene) which contains:

- Two different types of **Resource Pile** (objects on a pallet), which both produce resource items at a speed of 0.5 per second (/s)

- Two kinds of **Unit:**
    - A **Unit** (worker), which can move but has no further functionality
    - A **Transporter Unit** (forklift), which the user can set to transport resource items from a Resource to a predefined **base** (the space in front of the truck marked with a red circle)

Basic simulation functionality

In the basic simulation, the user can:

- Use arrow keys (or WASD) to move the camera around the warehouse space.
- Left-click any defined object to select it and open a UI overlay with its details
- When a Unit is selected, right-click:
    - Any location, to move it there
    - A Resource Pile, to set it to transport resource items from the object to the Base

Additional simulation requirements

As you work on this project, you will:

- Create a new **Productivity Unit** that increases resource item production in a base, and apply it to the worker in the warehouse
- Refine and optimize the code we have created for you, applying principles of object-oriented programming

Project styling

We've themed this project around a warehouse, in a lightweight low poly style. If you want to customize the project theme as you work on this functionality, you can import your own assets and swap them out.

Junior Programmer: Resource management simulation

Provided scripts

The project comes with seven main scripts which have been partially or completely written. There are also some additional scripts in subfolders that you can review but those aren't included here.

Base.cs

This is a subclass of the Building class**.** It doesn't change the behavior of that class, but it does store a singleton reference to it in its Awake function. This means that you can query the base from anywhere. This is used by Units when they return to the base.

Building.cs

This is an abstract base class of two subclasses: ResourcePile and Base. It keeps an inventory of what the class members contain, and also supports adding and removing items from that inventory. It also implements IUIInfoContent, the basic UI overlay in the simulation.

ColorHandler.cs

This script creates color buttons so that users can select a color to apply to the Transporter Units from the starting menu.

ResourcePile.cs

This is a subclass of the Building class**.** It takes a reference to a ResourceItem it will produce, and stores the following information about it in a Serializable class:

- name
- unique id
- associated sprite

It defines an Update function and increments a production counter by the production speed/s each update. Once this is above 1, it creates new resources and decrements the counter.

Note: It also overrides the GetData function for the IUIInfoContent of Building to give to the InfoPanel its production speed as data.

TransporterUnit.cs

This is a subclass of Unit. It contains:

- The target (Resource Pile) that a Transporter Unit is currently set to
- The amount of resources it can carry at once
- What is currently being transported by the Transporter Unit

The TransporterUnit subclass overrides three functions of its base class:

- How it behaves around Buildings in range
- Two GoTo functions, to react when the Transporter Unit's target changes

Junior Programmer: Resource management simulation

Unit.cs

This is an abstract base class for all Units. It controls Unit:

- Movement, including a target point for it to move to (which can be a Building or a position)
- Color, when this has been configured
- Behavior when in the range of its targeted Building

UserControl.cs

This class handles user input. It moves the Camera on the xz plane when reading vertical and horizontal values, and also controls left- and right-click functionality.