COS 214 Project: City Builder

Date Issued: 7 October 2024
Date Due: 4 November 2024
Submission Procedure: Upload via ClickUP
Submission Format: Archive (zip or tar.gz) containing your code, diagrams and pdfs.

1 Introduction

Urbanisation has been a defining trend in human civilisation, shaping societies, economies, and the environment. Managing the growth and sustainability of a city involves complex decision-making processes that balance infrastructure development, resource allocation, citizen satisfaction, and governance. Modern city builders not only focus on constructing buildings but also on creating efficient systems that ensure the well-being of its inhabitants and the prosperity of the urban environment.

In this project, students are tasked with developing a **City Builder Simulation** in C++. The simulation will require the application of various design patterns to ensure the system is scalable, maintainable, and flexible. By managing elements such as buildings, utilities, transportation, citizens, and government functions, students will demonstrate their ability to design and implement complex software systems using object-oriented principles and design patterns.

1.1 Objectives

The objectives of this project are to:

- Identify and analyse the functional requirements of a city-building simulation.
- Translate these requirements into a software design using appropriate design patterns.
- Develop UML diagrams to represent the system design and interactions.
- Implement the designed system in C++ with clear separation of concerns and modularity.
- Compile documentation detailing the design, implementation, and usage of the simulation.

- Collaborate effectively within a development team, utilising version control and best coding practices.

1.2 Outcomes

Upon successful completion of this project, students will:

- have hands-on experience in applying design patterns to solve software design problems.
- be able to work collaboratively in a team setting, managing codebases using Git.
- be able to produce good documentation detailing their system
- Be ready for COS 301

---

2 Constraints

1. **Team Composition:** Students must work in teams of 5 to 7 members.
2. **Design Patterns:** The project must incorporate and implement at least **10 design patterns**, focusing on aspects such as infrastructure, government systems and overall city management.
3. **UML Documentation:** Comprehensive UML diagrams must be provided, illustrating system design and complex interactions.
4. **Version Control:** Git must be used as the version control system, with all team members contributing regularly.
    - You will be required to join the COS214 Project organisation on GitHub. More information regarding this will be communicated via ClickUp.
5. **Documentation Generation:** Doxygen must be utilised to produce API documentation for the project.

---

3 Timeline

Your team will be expected to meet with your assigned tutor/AL weekly. This will ensure that your team stays on track to meet the deadline. The initial design component of the project also serves as practical 6.

1. **13 October** - Team registration of 5 to 7 members to be completed. Assign a team lead. Complete this registration on the CS portal
2. **15 October** - Submit initial project design (Practical 6)
3. **15 and 16 October** - Get initial design marked and receive feedback
4. **22 and 23 October** - Demonstrated updated initial design (if necessary) and skeleton implementation
5. **29 and 30 October** - Review minimal viable project and receive feedback
6. **4 November** - Submission of project on ClickUP.
7. **5 and 6 November** - Project demos.

Please do not fall behind schedule. The project carries a significant weight in your final mark and is very good preparation for your third year team-based modules.

---

4 Mark Allocation

Task	**Weights**
Practical Assignment 6	0
Design	30
Implementation	30
Report	15
Development Practices	10
Demo & Presentation	15
TOTAL	**100**

Note: Practical 6 has a 0 weight in terms of your final Project mark, but still contributes as a normal practical

4.1 Practical 6 Mark Allocation

Task	**Weights**
Functional Requirements	5
UML Class Diagram	15
UML State Diagram(s)	10
UML Activity Diagram(s)	10
UML Sequence Diagram(s)	10
UML Communication Diagram(s)	5
UML Object Diagram(s)	5
TOTAL	**50**

---

5 Project Description

In this project, you are required to develop a **City Builder Simulation** that models the intricacies of urban development and management. The simulation should allow players to construct and manage various aspects of a virtual city, ensuring sustainable growth and citizen satisfaction. The application of design patterns is crucial to handle the complexity and ensure that the system remains flexible and maintainable as it scales.

5.1 Simulation Engine

The core of the simulation is the engine that manages the dynamic interactions between different city components. The engine should handle events such as construction, resource allocation, taxation, and citizen needs in a real-time or turn-based manner.

Example Pseudocode for Simulation Loop:

```
public void simulationLoop() {
    while (simulationIsActive) {
        for (auto& sector : citySectors) {
            sector.update();
        }
        handleCitizenNeeds();
        manageResources();
        evaluatePolicies();
    }
}
```

Note: This pseudocode is a simplified example representation and should be expanded to accommodate all simulation aspects.

5.2 Components of the Simulation

This section proceeds to list some of the components of a city. However you are encouraged to identify additional components for your city builder simulator.

5.2.1 Buildings

The simulation should include various types of buildings, each serving different purposes:

- **Residential:** Houses, flats/apartments, townhouses, estates.
- **Commercial:** Shops, offices, malls.
- **Industrial:** Factories, warehouses, plants.
- **Landmarks:** Parks, monuments, cultural centers.

Each building type should have specific attributes and behaviours, influencing factors like citizen satisfaction, economic growth, and resource consumption.

5.2.2 Utilities

Utilities are essential for the functioning of the city. The simulation must model:

- **Power Plants:** Generate electricity to supply the city.
- **Water Supply:** Ensure a steady distribution of water.
- **Waste Management:** Handle waste removal and recycling.
- **Sewage Systems:** Manage sewage disposal and treatment.

Utilities should interact with buildings and citizens, affecting their functionality and satisfaction.

5.2.3 Transportation

Efficient transportation systems are vital for city mobility and economic activity. The simulation should include:

- **Roads:** Basic infrastructure for vehicle movement.

- **Public Transit:** Buses, taxis, etc.
- **Trains:** Freight and passenger trains.
- **Airports:** Facilitate air travel and cargo transport.

Transportation systems should impact traffic flow, commute times, and overall city connectivity.

5.2.4 Citizens

Citizens are the lifeblood of the city, driving demand for housing, employment, and services. The simulation should model:

- **Population Growth:** Dynamic increase in population based on various factors.
- **Employment:** Job availability influenced by industrial and commercial buildings.
- **Services:** Healthcare, education, security, and entertainment.
- **Satisfaction:** Affected by factors like taxes, amenities, and quality of life.

Citizens should respond to government policies, economic changes, and infrastructural developments.

5.2.5 Government

The government system manages city governance, including:

- **Taxation:** Setting and collecting taxes from citizens and businesses.
- **City Budget:** Budget allocation for various city services and projects.
- **Policies:** Implementing laws and regulations that impact city dynamics.
- **Public Services:** Managing healthcare, education, law enforcement, and more.

Government decisions should influence citizen satisfaction, economic growth, and overall city development.

5.2.6 Resources

Managing resources is critical for sustaining city operations. The simulation should track:

- **Materials:** Construction resources like wood, steel, and concrete.
- **Energy:** Power generation and consumption.
- **Water:** Supply and distribution.
- **Budget:** Financial resources for city management and development.

Efficient resource management should enable city expansion and the provision of services.

5.2.7 Taxes

Taxation is a key mechanism for funding city services and infrastructure. The simulation should model:

- **Tax Rates:** Adjustable rates for different categories (e.g., income, property, sales).
- **Collection:** Mechanisms for tax collection from citizens and businesses.
- **Allocation:** Distributing collected taxes to various city departments and projects.
- **Impact:** Effects of tax changes on citizen satisfaction and economic activity.

5.2.8 City Growth

City growth should be a dynamic process influenced by multiple factors:

- **Population Growth:** Driven by birth rates, migration, and economic opportunities.
- **Housing Needs:** Expansion of residential buildings to accommodate growing population.
- **Economic Development:** Growth of commercial and industrial sectors leading to more jobs.
- **Infrastructure Expansion:** Development of utilities and transportation systems to support growth.

Growth mechanics should ensure a cascading effect where each aspect of the city influences others, creating a realistic simulation of urban development.

6 Tasks

Task 1: Practical Assignment 6

For Practical Assignment 6 you are required to submit initial UML diagrams to demonstrate design. Use the steps outlined in Task 2 to help you complete this.

The Practical 6 Mark Allocation in Section 4 outlines the requirements for this submission.

The due date for practical 6 is 15 October. In the practical sessions on the 15th and 16th of October you will be required to present your initial design to your assigned tutor/AL. They will mark it (as your practical 6 mark) and provide feedback. This feedback needs to be incorporated in your final design to be marked with the rest of your project.

Tip from someone who has been there:

DO NOT just split the work and design separately, eg. member A you do the class diagram and member B you do the functional dependencies. Identify functional dependencies together, and design the high-level processes using activity diagrams. Design the rough overall class diagram together as a team as well. This ensures everyone is on the same page. The small details and more granular diagrams can then be split among members. Have regular team feedback sessions to ensure that there are no misconceptions

Task 2: Design (30 marks)

Task 2.1 Identify the Functional Requirements

- Define the core functionalities of the city builder simulation.
- Detail the interactions between different components such as buildings, utilities, transportation, citizens, and government.
- Subsystems are a good way to split of functional requirements and organise your thoughts.

Task 2.2 Design the Processes Using Activity Diagrams

- Create activity diagrams that illustrate the workflows for key processes like construction, resource allocation, taxation, and citizen satisfaction management.
- We suggest you also create a high level activity diagram illustrating the overall flow

Task 2.3 Select Design Patterns to Address Functional Requirements

- Identify and justify the selection of at least **10 design patterns** from Gamma et al.'s catalog.
- Examples may include Singleton for resource management, Observer for citizen satisfaction updates, Factory for building creation, etc.
- You need to include at least 4 of the design patterns introduced after Chapter 13 (`https://www.cs.up.ac.za/cs/lmarshall/TDP/TDP.html`):
  - Observer
  - Iterator
  - Mediator
  - Command
  - Adapter
  - Chain of Responsibility
  - Builder
  - Interpreter
  - Bridge
  - Facade
  - Visitor
  - Proxy
  - Singleton
  - Flyweight

Task 2.4 Design the Classes for Identified Patterns

- Develop class structures that implement the chosen design patterns.
- Define the relationships and interactions between classes, ensuring adherence to design principles.

Task 2.5 Draw a Class Diagram of Your System

- Provide a comprehensive UML class diagram that showcases all classes, their attributes, methods, and relationships.

Task 2.6 Draw Sequence and Communication Diagrams

- Illustrate the message passing and interactions between objects for key scenarios within the simulation.

Task 2.7 Design State Diagrams

- Create state diagrams for objects that undergo state changes, such as buildings transitioning from under construction to operational, or citizens changing their satisfaction levels.

Task 2.8 Provide Object Diagrams

- Include at least two object diagrams that depict the state of active objects in the simulation at specific points in time.

Task 3: Implementation (30 marks)

Task 3.1 Implement the City Builder Simulation

- Develop the simulation in C++, ensuring all functionalities are operational.
  - Develop your code based on a contract-first approach, defining interfaces and contracts before implementation.
  - .h files work well for defining contracts and interfaces.
  - This ensures team members can work on dependent systems before completion of the other.
- Ensure correct separation of concern.
- Implement at least a text-based interface that allows players to interact with the simulation, such as adding/removing buildings, managing resources, and setting tax rates.

Task 3.2 (Optional - Bonus) Implement a Graphical Interface

- While not required, developing a graphical user interface (GUI) will provide bonus marks and enhance the user experience.
- GUIs can be developed in a C++ GUI framework or as a website (with you C++ engine acting as the API). You may use any libraries.

Task 4: Report (15 marks)

4.1 Research Brief

- Write a short brief (1 page) on urban development, city management principles, and the role of various components within a city.

- Reference all sources appropriately.
- Document how these components influenced your design
- Document your assumptions, design decisions, and provide relevant definitions and explanations.

4.2 Design Pattern Application Report

- Detail how each of the chosen design patterns has been applied to address specific functionalities within the system.
- Augment explanations with relevant UML diagrams to illustrate the implementation of design patterns. This is not limited to UML class diagrams.

Task 5: Development Practices (10 marks)

5.1 Version Control with Git

- Use Git as the Version Control System.
- Document your branching strategy in the pdf report.
- Ensure every team member makes at least 15 commits, documenting significant changes and updates.
- Ensure at least 3 closed pull requests

5.2 Code Documentation

- Document your code following C++ documentation standards.
- Use meaningful comments and descriptions for classes, methods, and complex logic.
- Use Doxygen to produce comprehensive documentation.
- Ensure all public classes and methods are properly annotated for Doxygen processing.

5.3 Automated Unit Testing

- Implement unit and/or integration tests using an automated testing framework.
- While full coverage is not mandatory, ensure that key functionalities are tested.
- Every team member must contribute to the creation of unit tests.

5.4 (Optional - Bonus) Github Actions Linter and Tester

- While not required, develop and deploy CI/CD pipelines to lint, test (using your automated unit tests) and build your application using Github Actions)

Task 6: Demo & Presentation (15 marks)

- **Team Participation:** All team members must be present during the demo. Failure to attend will result in penalties.
- **Professionalism:** The demo should be well-prepared, demonstrating all key features and functionalities of the simulation.
- **Clarity:** Present your design decisions, implementation challenges, and how design patterns were utilised effectively.
- **Engagement:** Engage the audience with clear explanations, visual aids, and a live demonstration of the simulation in action.
- **Key Points:**
  - Present a brief overview of your system
  - Present your class diagram and any other diagrams deemed necessary to understand the design of your system
  - Demonstrate your simulation. It is important that you explain/show which design patterns are being used in the different processes.
  - Show your github (commit graph, branches and pull requests - screenshots are perfect)
  - Show your doxygen documentation
  - Show your tests
  - Code should not be shown during the demo

---

7 Submission Instructions

Each team is required to create a Git repository to manage the project. Use GitHub to host your repository. Ensure that the documentation includes a link to your GitHub repository. The repository must contain the following:

- **System Files:** All source files (`.h` and `.cpp`) and your Makefile.
- **Data Files:** Any necessary data files required to run the program.

- **Readme:** A `readme.md` in the root directory containing a short description of your project, explaining how to compile and run the program, and the placement of any data files.
- **Report:** A PDF version of your latest report in a folder named `Report`. The report must also be written in Google Docs, with a link to the Google Docs version included in the PDF.
- **System Folder:** Place all system files in a folder named `System`. Running your Makefile in this folder should compile and link your system correctly.
- **Data Folder:** Place your data files, if any, in a folder named `Data`.

Submission Steps:

1. Ensure your Git repository is complete with all required files and folders.
2. Download your git repository
3. Archive the repository as a `.zip` or `.tar.gz` file.
4. Upload the archive to the ClickUP submission slot before the deadline.
5. Ensure the team lead uploads the file to avoid penalties.

Note: Failure to upload the project to ClickUP will result in the team receiving a score of 0. It is the responsibility of the team lead to ensure timely submission.

Demo Scheduling:

You will be required to demonstrate your system during the week of 5 November. Ensure all team members are prepared and available to present.
