
DESIGN WITS CITY GAME SIMULATION
Your City, Your Legacy

Functional Requirements

The City Simulator will:

Provide a City Manager:

- Select simulation modes
- Initialize city zones and layouts
- Control resource allocation to buildings
- Let players make economic decisions
- Allow players to inspect buildings and zones

Provide Citizens that:

- Have unique identifiers and demographics
- Can be employed in buildings
- Pay taxes based on income
- Have variable satisfaction levels
- Consume city resources
- Use transport networks

Provide Buildings that:

- Can be constructed and have different types such as:
    - Commercial Buildings
    - Residential buildings

    - Industrial Buildings
    - Public Service Buildings
    - Landmark Buildings
- Require maintenance and resources
- House citizens (residential)
- Provide employment (commercial/industrial)
- Generate tax revenue
- Have capacity limits
- Affect citizen satisfaction based on their creation and change in state
- Form the city infrastructure

Provide a Government that:

- Collects taxes from citizens
- Manages city budget
- Implements city policies
- Responds to city needs
- Maintains infrastructure

Provide Resources that:

- - Supply essential utilities:
    - Water
    - Power

    - Waste Management

- Have production/con
- Require storage facilities
- Need distribution to the buildings
- Affect building operations ● Impact citizen satisfaction

Provide a Finance Department that:

- Manages tax collection
- Is used by the government to manage the finances and funds of the city
- Handles fund allocation different components

Provide a Transport Manager that:

- Support different vehicle types
- Manage route systems
- Handle capacity allocation Provide a Utilities Manager that:

- Controls resource production
- Manages distribution of resources
- Monitors resource levels

Diagrams

Research Brief Report

Urban development is the planning and management of land use in cities to create sustainable and efficient environments. This process involves considering various factors such as housing, transportation, infrastructure, and public services. City management principles guide urban development by emphasizing the integration of key components that contribute to a city's functionality and quality of life.

City Management Principles

Effective city management relies on various principles, including:

- **Sustainability:** Balancing economic, social, and environmental needs.
- **Equality:** Ensuring all citizens have access to services and opportunities.
- **Efficiency:** Utilizing resources wisely to maximize public benefit.
- **Adaptability:** Preparing for and adapting to changes and challenges.

Components of Urban Development

Key components influencing urban design include:

- **Infrastructure:** Roads, bridges, and public transport systems that facilitate mobility.
- **Economic Development:** Focuses on job creation and business growth to sustain the economy.
- **Public Spaces:** Parks and recreational areas that enhance quality of life.
- **Residential Areas:** Provide housing for citizens and promotes community interaction.

Assumptions and Design Decisions

Assumptions made during the design process include:

- Citizens value proximity to essential services and employment.
- Improved transport networks that reduce congestion.
- A balanced mix of residential and commercial zones leads to economic vitality.

Understanding urban development and city management principles is crucial for creating sustainable and vibrant urban spaces. Each component plays a vital role in shaping the city's future.

References

1. ClearPoint Strategy. "Types of Urban Planning Explained." *ClearPoint Strategy Blog*, 2023. [Online]. Available: https://www.clearpointstrategy.com/blog/types-of-urban-planning
2. Metropolis. "Principles for Better Cities." *Metropolis*, 2020. [Online]. Available: https://www.metropolis.org/sites/default/files/resources/Principles-for-Better-Cities.pdf

Design Pattern Application Report

State Design Pattern

The State pattern enables a Building to modify its behavior through changeState() and handle() functions based on its current state (UnderConstruction, Damaged, or Operational), where each state class implements specific behavior for common operations, making the Building appear to change its class when transitioning between states.

Class	**Participant**
Building	Context
State	State
UnderConstruction , Operational , Damaged	Concrete State

Abstract Factory Design Pattern

The Abstract Factory pattern is applied to manage building creation through a hierarchy of specialized factories. The Building (AbstractFactory) interface defines the common creation method (createBuilding), while concrete factories (ResidentialFactory, CommercialFactory, IndustrialFactory, LandmarkFactory, PublicServiceFactory) handle the creation of their specific building types. This pattern ensures that buildings are created consistently within their categories (e.g., houses through HouseFactory under ResidentialFactory), enforces zoning rules, and maintains clean separation between different building types while allowing easy addition of new building categories without modifying existing code.

Class	**Participant**
ResidentialFactory , IndustrialFactory , CommercialFactory , PublicServiceFactory , LandmarkFactory	AbstractFactory
HouseFactory , ApartmentFactory , TownhouseFactory , FactoriesFactory , PowerplantFactory , WarehouseFactory , ShopFactory , MallFactory , OfficeFactory , SchoolFactory , PoliceStationFactory , MedicalCenterFactory , ParkFactory , MonumentFactory , CulturalCenterFactory	ConcreteFactory
Residential , Industrial , Commercial , PublicService , Landmark	AbstractProduct
House , Apartment , Townhouse , Factories , Powerplant , Warehouse , Shop , Mall , Office , School , PoliceStation, MedicalCenter , Park , Monument , CulturalCenter	ConcreteProduct

Factory Method Design Pattern

The Factory Method pattern has been implemented to manage two crucial aspects of city infrastructure. First, it handles basic utility creation through the UtilityFactory hierarchy, where specialized factories create specific infrastructure components - PowerPlantFactory for power generation facilities, WaterFactory for water supply systems, WasteFactory for waste management facilities, and SewageFactory for sewage treatment plants.

Class	**Participant**
UtilityFactory	Creator
PowerPlantFactory , WaterFactory , WasteFactory ,SewageFactory	ConcreteCreator
Utility	Product
UtilityPowerPlant , WaterSupply, WasteManagement ,SewageSystems	ConcreteProduct

Second, it manages energy source diversity through the EnergyFactory hierarchy, with specialized factories (HydroFactory, WindFactory, NuclearFactory, and CoalFactory) creating different types of power generation facilities to meet the city's varying energy needs.

Class	**Participant**
EnergyFactory	Creator
HydroFactory , WindFactory ,NuclearFactory , CoalFactory	ConcreteCreator
EnergySource	Product
HydroSource , WindSource ,NuclearSource , CoalSource	ConcreteProduct

Command Design Pattern

The Command pattern functions within two vital control systems in the city simulation. The utility control system uses UtilityManager as an invoker to execute StartCommand and StopCommand operations, managing the activation and deactivation of various utility services through the Utility receiver.

Class	**Participant**
UtilityManager	Invoker
Command	Command
StartCommand , StopCommand	ConcreteCommand
Utility	Receiver

The government policy system employs the Government class as an invoker to implement PublicServicePolicies and EconomicPolicies, with these commands directly affecting the city's population through the CitizenInterface receiver. This dual implementation provides flexible control over both infrastructure operations and policy implementation.

Class	**Participant**
Government	Invoker
Policies	Command
PublicServicePolicies , EconomicPolicies	ConcreteCommand
CitizenInterface	Receiver

Prototype Design Pattern

The Prototype pattern serves the city's need for rapid infrastructure expansion and population growth. In the utility sector, it enables quick cloning of existing infrastructure including UtilityPowerPlant, WaterSupply, WasteManagement, and SewageSystems, allowing for efficient expansion of city services.

Class	Participant
Utility	Prototype
UtilityPowerPlant , WaterSupply, WasteManagement ,SewageSystems	ConcretePrototype

In the population management aspect, it facilitates the creation of new citizens by cloning existing Citizen and CitizenType templates, streamlining the process of population growth and demographic diversification.

Class	Participant
CitizenInterface	Prototype
Citizen , CitizenType	ConcretePrototype

Chain of Responsibility Design Pattern

The Chain of Responsibility pattern manages two key processing chains within the city. The route management system uses RouteNode as a base handler with BestRouteNode and AccessibleRoute concrete handlers to optimize city navigation and ensure accessible pathways.

Class	Participant
RouteNode	Handler
BestRouteNode , AccesibleRoute	ConcreteHandler

The government operations chain, built with Government as the base handler and specialized processors like UtilitiesSector, FinanceSector, and GeneralSector, efficiently routes citizen requests through appropriate departments, ensuring each concern is addressed by the most suitable authority.

Class	Participant
Government	Handler
UtilitiesSector , FinanceSector, GeneralSector	ConcreteHandler
CitizenInterface	Client

Iterator Design Pattern

The Iterator pattern has been specifically applied to the city's transportation network, using RouteNode as the aggregate interface with BestRouteNode and AccessibleRoute as concrete implementations. The MapIterator provides systematic navigation through the city's route network, enabling efficient pathfinding and transportation planning while keeping the underlying route structure encapsulated.

Class	Participant
RouteNode	Aggregate
BestRouteNode , AccesibleRoute	ConcreteAggregate
MapIterator	Iterator / ConcreteIterator

Strategy Design Pattern

The Strategy pattern addresses the city's transportation needs through the TravelManager context, which coordinates different transport strategies including Train, Vehicle, and Plane. This implementation allows citizens to seamlessly switch between different transportation methods while maintaining consistent travel management interfaces throughout the city.

Class	Participant
TravelManager	Context
Transport	Strategy
Train ,Vehicle , Plane	ConcreteStrategy

Builder Design Pattern

The Builder pattern manages complex transport system creation through the TravelManager director and TransportBuilder interface. Specialized builders (PlaneBuilder, TrainBuilder, VehicleBuilder) handle the intricate process of creating different transportation options, ensuring each transport type is constructed correctly and consistently while maintaining flexibility in the construction process.

Class	**Participant**
TravelManager	Director
TransportBuilder	Builder
PlaneBuilder , TrainBuilder , VehicleBuilder	ConcreteBuilder
Train , Vehicle , Plane	Product

Mediator Design Pattern

The Mediator pattern centralizes resource management through the ResourceMediator, which coordinates interactions between various city resources including power, water, revenue, waste management, and sewage systems. This central coordination point simplifies complex resource interdependencies and ensures efficient resource allocation throughout the city.

Class	**Participant**
ResourceMediator	Mediator / ConcreteMediator
Resources	Colleague
powerResource ,waterResource , revenueResource , wasteManagementResource , sewageManagementResource	ConcreteColleague

Observer Design Pattern

The Observer pattern has been specifically implemented to handle the city's tax collection notification system. The Government class acts as the subject (publisher) that monitors when tax collection periods begin, while the CitizenInterface serves as the observer interface. Each Citizen, as a concrete observer, is automatically notified by the Government when it's time to collect taxes. This pattern ensures efficient tax collection by automatically informing all registered citizens when they need to pay their taxes, eliminating the need for manual notification systems or periodic checks. The implementation streamlines the tax collection process by maintaining a direct communication channel between the Government and all citizens for tax-related matters.

Class	**Participant**
Government	Subject / ConcreteSubject
CitizenInterface	Observer
Citizen	ConcreteObserver

Decorator Design Pattern

The Decorator pattern enhances citizen functionality by using CitizenInterface as the base component and adding specialized behaviors through CitizenType decorators. The concrete decorators EmployedCitizen and PropertyOwner add specific responsibilities and privileges to citizens, allowing for dynamic modification of citizen capabilities based on their roles in the city.

Class	**Participant**
CitizenInterface	Component
Citizen	ConcreteComponent
CitizenType	Decorator
EmployedCitizen ,PropertyOwner	ConcreteDecorator

Facade Design Pattern

The Facade pattern simplifies access to complex subsystems through two main facades. The finance system uses FinanceDepartment to provide a unified interface to various taxation and budget systems.

Class	Participant
Government	Client
FinanceDepartment	Façade
CommercialTaxationSystem , ResidentialTaxationSystem , BudgetAllocationSystem	Subsystem

The city control system employs CityController to coordinate CitizenController, BuildingController, and UtilitiesController. These facades reduce system complexity and provide clear access points for managing various city operations.

Class	Participant
CityController	Façade
CitizenController , BuildingController , UtilitiesController	Subsystem