Architecture

Cohort 2- Group 1

Members

Ollie Barlow (vrq506)
Jacob Barnes (bdp514)
Zayaan Bawany (fst508)
Lily Boldbaatar (jkk522)
Georgia Briggs (rxm503)
Evelyn Gravett (qnf502)
Hassan Yunus (wfv505)

Architecture Diagrams

Our architecture is represented with the behavioural and structural diagrams pictured below, developed in UML. We used PlantUML to create these diagrams as it allowed us to present the architecture in a clear and readable way.

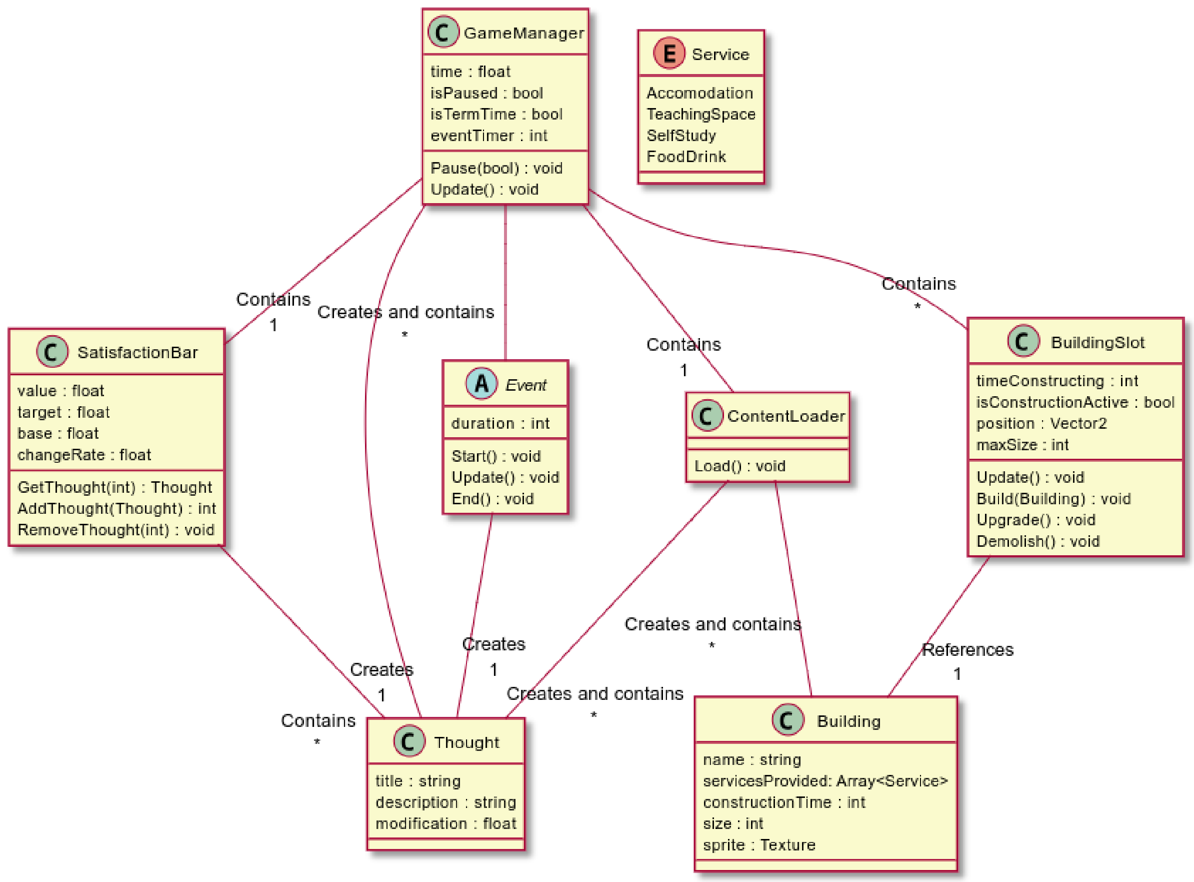
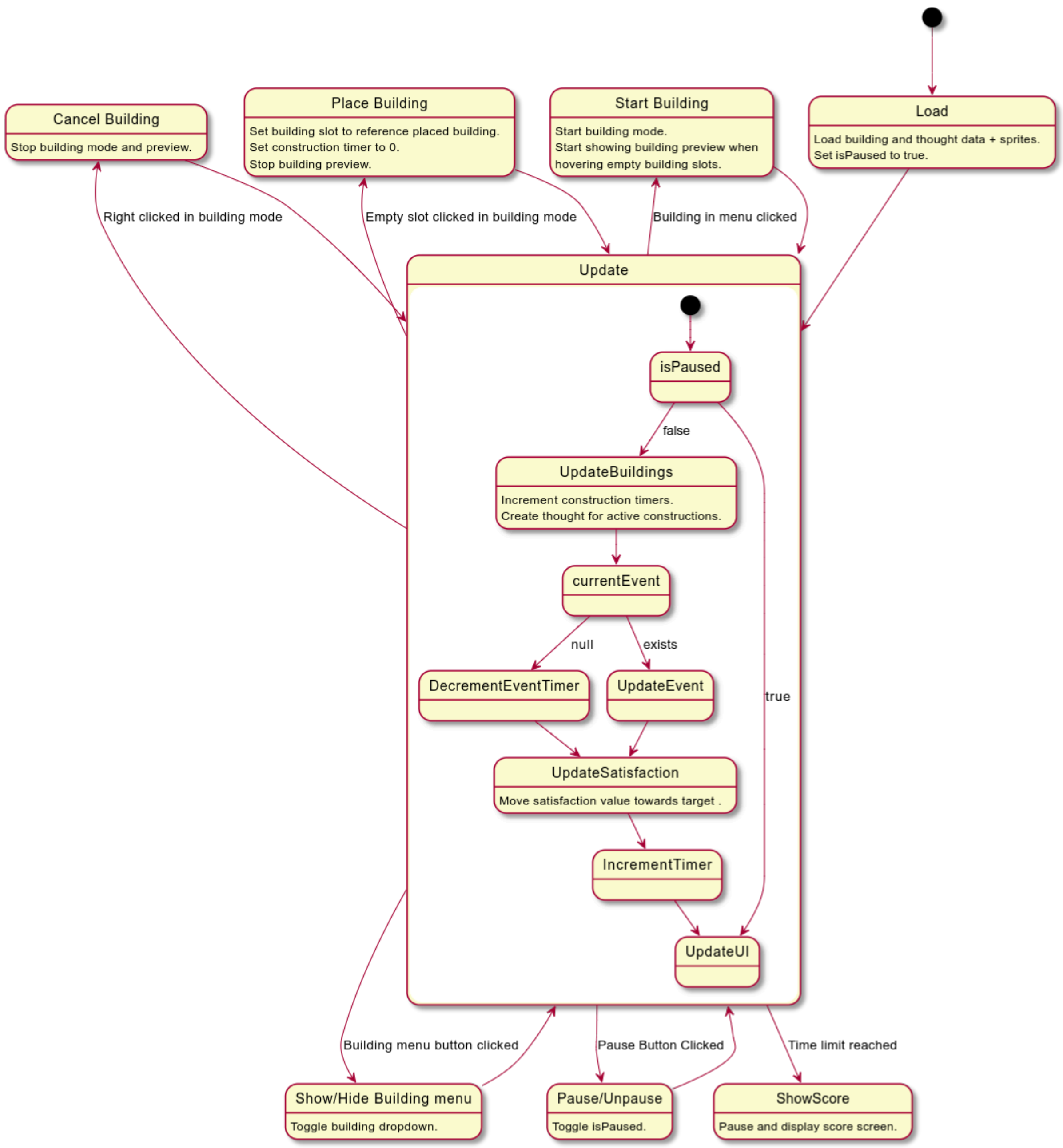


Diagram 1- UML Class diagram

Represents the structure of the system, details classes and their relationships (such as which classes they may create or contain) as well as their attributes and methods.



Cancel Building
Stop building mode and preview.
Place Building
Set building slot to reference placed building.
Set construction timer to 0.
Stop building preview.
Start Building
Start building mode.
Start showing building preview when hovering empty building slots.
Load
Load building and thought data + sprites.
Set isPaused to true.
Right clicked in building mode
Empty slot clicked in building mode
Building in menu clicked
Update
isPaused
false
UpdateBuildings
Increment construction timers.
Create thought for active constructions.
currentEvent
null
exists
DecrementEventTimer
UpdateEvent
true
UpdateSatisfaction
Move satisfaction value towards target .
IncrementTimer
UpdateUI
Building menu button clicked
Pause Button Clicked
Time limit reached
Show/Hide Building menu
Toggle building dropdown.
Pause/Unpause
Toggle isPaused.
ShowScore
Pause and display score screen.

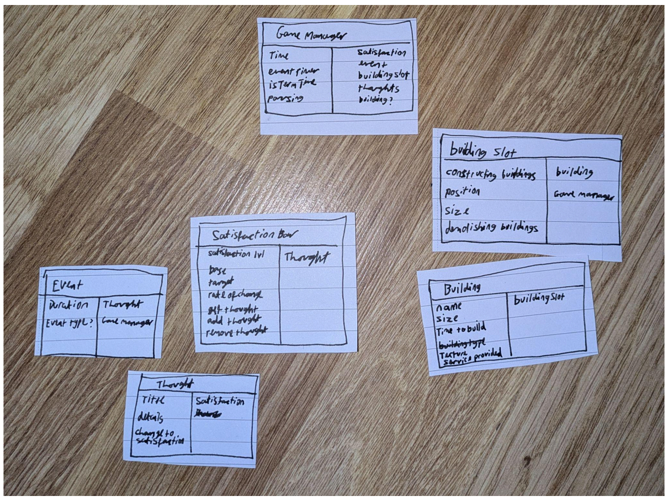
Diagram 2- UML State machine diagram

Represents the states a system can be in and the transitions between these states (e.g. Pausing the game would change ‘isPaused’ to true).

Justification for the Architecture

Our architecture has a modular approach which allows for maintainability and future extensions to the project as each class manages its own specific area such as the satisfaction bar or buildings meaning adding new features to the project, such as building upgrades or new building types, would require minimal change to the existing code. Debugging is also made easier by this approach as errors with specific features can be found quickly as it'd relate to only one or two classes.

Below is an initial sketch of some CRC cards, created by deciding the core components that make up the game and then what each of them manages and the relationships between them. This has since evolved, for example the ContentLoader class was added later on.



Relation to requirements

Behavioural diagram

The diagram includes states 'Cancel Building', 'Place building' and 'Start building' linking to requirement UR_BUILDINGS giving the user the ability to create and cancel buildings as well as 'Update Building' giving the user information on construction time left,as well as changing satisfaction levels based on construction. There is an UpdateUI state relating to the UR_UI requirement to ensure the user interface is clear and up to date. The Pause/unpause state meets the UR_PAUSE requirement as it allows the user to pause at any time and the 'update' is constantly checking as to whether the game has been paused. The 'Show Score' state meets the UR_SATISFACTION_METRE requirement as it displays the current satisfaction to the user as well as 'Update satisfaction' keeping this information up to date. The state 'Increment Timer' meets the requirements UR_DISPLAY_TIMER and NFR_TIMER_VISIBILITY as it keeps track of the time which will be displayed to the user. There are multiple states for events: currentEvent, DecrementEventTimer and UpdateEvent which relate to FR_RANDOM_EVENT and FR_SET_EVENT as these keep track of what events are happening and how long for.

Structural Diagram

BuildingSlot:
Relates to requirement FR_BUILDING_OVERLAP and FR_BUILDING_PLACEMENT as we created set slots available for the user to build in that once filled cannot contain multiple buildings. This class contains methods ‘Build’,’Upgrade’ and ‘Demolish’ referenced in the behavioural diagram that meet requirement UR_BUILDINGS. The attributes timeConstructing and isConstructionActive will allow the system to meet the requirements FR_CONSTRUCTION_SATISFACTION_LOSS and FR_BUILDING_CONSTRUCTION as it keeps track of how long construction is happening for to adjust the satisfaction loss to only be within that time frame and allow gain in satisfaction once completed.

SatisfactionBar/ Thought:
Contains attributes ‘value’, ‘target’ and ‘base’ which meet requirement UR_SATISFACTION_METRE as it keeps track of satisfaction level to display to the user. It also contains attribute ‘changeRate’ meeting requirement NFR_SATISFACTION_MULT and UR_SATISFACTION_METRE as it contains the rate at which the satisfaction increases which can be affected by different things such as buildings etc. there must be a rate of change instead of just jumping up to a new value to help to user visibly see the changes happening. SatisfactionBar also has methods ‘GetThought’,’AddThought’ and ‘RemoveThought’ and contains class ‘Thought’ which represents thoughts students may have about current events or buildings- this represents changes to satisfaction bar to the user for example if the reason satisfaction is decreasing is due to lack of library space, a thought may pop up saying ‘I have nowhere to study!’ which meets the requirement NFR_SATISFACTION_BREAKDOWN.

GameManager:
Contains an attribute ‘isTermTime’ so that it can track changes to satisfaction as construction during term time results in a decrease to satisfaction, this meets requirement FR_CONSTRUCTION_SATISFACTION_LOSS. Contains attribute Time to keep track of the 5 minute game timer to meet requirement UR_DISPLAY_TIMER. Method Pause affects the attribute isPaused and is used to meet requirements UR_PAUSE, FR_SATISFACTION_PAUSE and FR_TIMER_PAUSE as it ensures no changes to satisfaction or the game timer while isPaused is true as well as the Pause method allowing the user to pause at any time.