Design Description - Team 10

Project2: Heated Earth – October 21, 2014

Timothy Laurent (tlaurent3)
Prabhakar Kallakuri (pkallakuri3)
John Swenson (jswenson8)

Overview

The heated Earth project is based on scientific concept of heat transfer from a pointlike energy source, the Sun, to a sphere orbiting on a non tilted axis at a constant distance, the Earth. The design document studies the four variants where each of four different design choices affect the program: having a master program control the execution of a data producer and a data consumer, having the data consumer control the data producer, having the data producer control the data consumer, and finally how the size of a buffer between producer and consumer affects system throughput.

The most challenging aspect of the design is to accommodate all the above variants in to a single graphical user interface. This required the designers to have a strong understanding of the principles of software design and factors in order to create a cohesive design. The structure of the design uses an interface to generalize the simulation and presentation types so that those that take the initiative and those that do not can be used interchangeably. This allows the driving threads to be agnostic to the underlying structure of the interaction between the presentation and simulation.

Program Characteristics:

Class	.Java File Size	.Class File Size	LoC	Operations	Attributes	Dependencies
Demo.Java	5313	4774	133			
SimBuffer	1131	1727	31			
SimulationCtl	3767	3284	138			
SimulationFactory	2305	2138	75			
SimulationConstants	859	747	23			
SimSettings	2404	2098	47			
SimCmdMsg	1536	1900	52			
SimCmdReceivable	459	296	8			
SimState	1120	868	34			
TempGrid	447	675	16			
DisplayPanel	393	670	11			
CmdListenable	120	169	5			
CmdListener	85	135	4			
ControlPanel	10539	7427	272			
EarthGridDisplay	5674	4697	117			
EarthImage	674	990	21			
EarthPanel	2982	2387	57			
SunDisplay	3036	2266	61			
TemperatureColorPicker	2119	1544	74			
TemperatureGrid	690	247	5			
Util	2220	973	16			

Presentation	1005	1205	27			
PresM	2309	2727	84			
PresS	1726	2564	63			
EarthCell	2912	1335	51			
ICell	1352	381	11			
SimM	1663	2154	54			
SimS	1684	2529	66			
SimulatedEarth	7519	5414	174			
Simulation	657	790	21			
EarthCellTests	2341	1468	64			
SimulatedEarthTestAccessor	1263	1825	49			

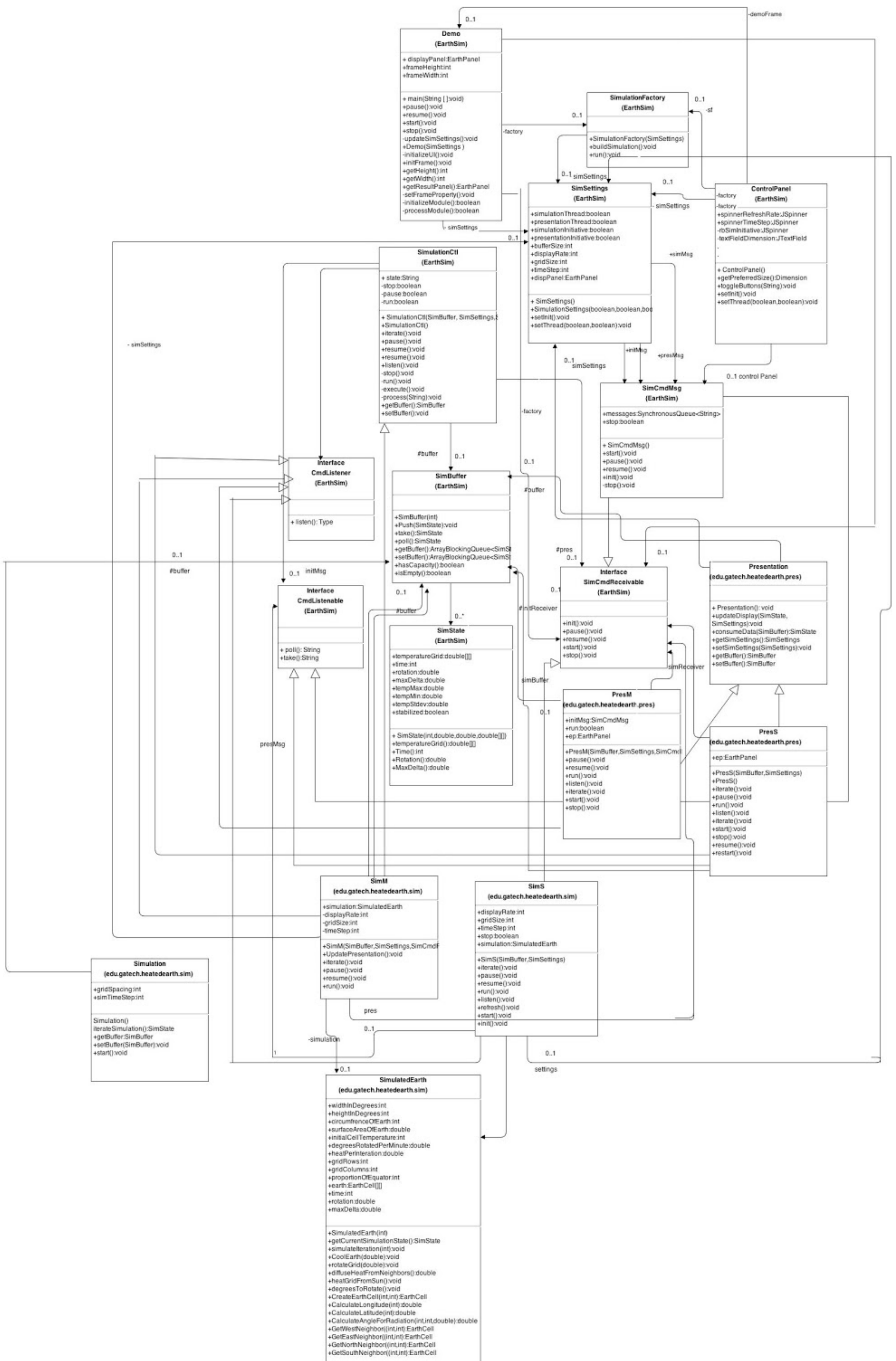
Static Description

Class Name	Roles in the Architecture	Dependencies
ICell	Interface provides the common methods for heatable cells	● None
EarthCell	Contains information about a cell on the Earth	● None
SimM	Runs the simulation code when the simulation has the initiative	● *calls* SimBuffer ● *instantiates* SimulatedEarth ● *uses* SimSettings *as a parameter* ● *implements* Simulation ● *calls* SimCmdReceivable

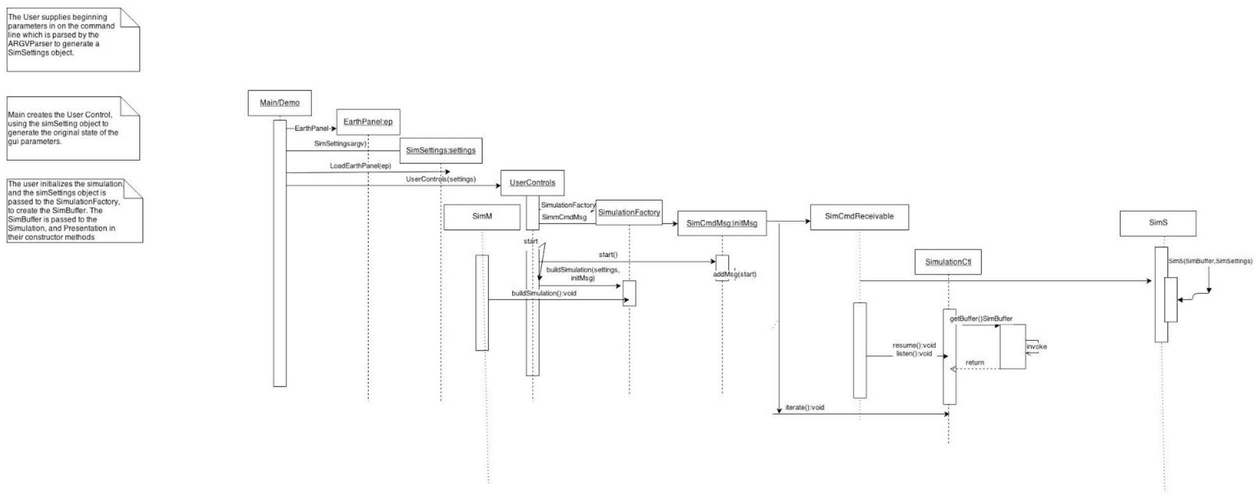
SimS	Runs the simulation code when the simulation does not have the initiative	● *calls* SimBuffer ● *instantiates* SimulatedEarth ● *uses* SimSettings *as a parameter* ● *calls* CmdListenable
SimulatedEarth	Provides calculations about the simulation of a heated earth.	● *instantiates* EarthCell
Simulation	Abstract class that provides functionally around simulating a heated Earth	●
Presentation	Abstract class that provides functionality around showing the results of the Simulated Earth	●
PresM	Handles the presentation of the simulation when the presenter has the initiative.	● *instantiates* EarthPanel ● *calls* SimBuffer ●
Demo.Java	Main driver for the program.	● *instantiates* EarthPlate ● *instantiates* CmdRecieveable
SimBuffer	Holds data in order to connect the simulation and presentation.	
SimulationCtl	Holds the initiative when neither the Simulation or Presentation are in control.	● *calls* SimCmdRecievable.start for both the simulation and presentation objects
SimulationFactory	Determines which types of simulations and presentations to create based on the simulation settings	● *instantiates* PresM, PresS, SimS, SimM, SimBuffer

SimulationConstants	Contains constants related to the simulation.	●
SimSettings	Holds information related to the current simulation	
SimCmdMsg	Transmits Commands	
SimCmdReceivable	Interface for an object that receives commands	
SimState	Holds information about the current temperature of the Earth and the position of the Sun	
TempGrid	Implementation of the TemeratureGrid interface used by the presenter	● *Implements* TemperatureGrid interface
DisplayPanel	Controls the display of the presentation to the user.	
CmdListenable	Interface for an object that can be listened to for commands	
CmdListener	Interface for an object that listens for commands	
ControlPanel	User Interface control for the application	● *Instantiates* SimulationFactory ● *Uses* Demo *as a Parameter.*

UML Class Model Diagram


Demo
(EarthSim)
+ displayPanel:EarthPanel
+frameHeight:int
+frameWidth:int
+ main(String [ ]:void)
+pause():void
+resume():void
+start():void
+stop():void
-updateSimSettings():void
+Demo(SimSettings)
-initializeUI():void
+initFrame():void
+getHeight():int
+getWidth():int
+getResultPanel():EarthPanel
-setFrameProperty():void
-initializeModule():boolean
-processModule():boolean
SimulationFactory
(EarthSim)
+SimulationFactory(SimSettings)
+buildSimulation():void
+run():void
SimSettings
(EarthSim)
+simulationThread:boolean
+presentationThread:boolean
+simulationInitiative:boolean
+presentationInitiative:boolean
+bufferSize:int
+displayRate:int
+gridSize:int
+timeStep:int
+dispPanel:EarthPanel
+ SimSettings()
+SimulationSettings(boolean,boolean,boo
+setInit():void
+setThread(boolean,boolean):void
ControlPanel
(EarthSim)
-factory
-factory
+spinnerRefreshRate:JSpinner
+spinnerTimeStep:JSpinner
-rbSimInitiative:JSpinner
-textFieldDimension:JTextField
+ ControlPanel()
+getPreferredSize():Dimension
+toggleButtons(String):void
+setInit():void
+setThread(boolean,boolean):void
SimulationCtl
(EarthSim)
+ state:String
-stop:boolean
-pause:boolean
-run:boolean
+ SimulationCtl(SimBuffer, SimSettings,
+SimulationCtl()
+iterate():void
+pause():void
+resume():void
+resume():void
+listen():void
-stop():void
-run():void
-execute():void
-process(String):void
+getBuffer():SimBuffer
+setBuffer():void
SimCmdMsg
(EarthSim)
+messages:SynchronousQueue<String>
+stop:boolean
+ SimCmdMsg()
+start():void
+pause():void
+resume():void
+init():void
-stop():void
Interface
CmdListener
(EarthSim)
+ listen(): Type
SimBuffer
(EarthSim)
+SimBuffer(int)
+Push(SimState):void
+take():SimState
+poll():SimState
+getBuffer():ArrayBlockingQueue<SimSt
+setBuffer():ArrayBlockingQueue<SimSt
+hasCapacity():boolean
+isEmpty():boolean
Interface
CmdListenable
(EarthSim)
+ poll(): String
+take():String
SimState
(EarthSim)
+temperatureGrid:double[][]
+time:int
+rotation:double
+maxDelta:double
+tempMax:double
+tempMin:double
+tempStdev:double
+stabilized:boolean
+ SimState(int,double,double,double[][])
+temperatureGrid():double[][]
+Time():int
+Rotation():double
+MaxDelta():double
Interface
SimCmdReceivable
(EarthSim)
+init():void
+pause():void
+resume():void
+start():void
+stop():void
Presentation
(edu.gatech.heatedearth.pres)
+ Presentation(): void
+updateDisplay(SimState, SimSettings):void
+consumeData(SimBuffer):SimState
+getSimSettings():SimSettings
+setSimSettings(SimSettings):void
+getBuffer():SimBuffer
+setBuffer():SimBuffer
PresM
(edu.gatech.heatedearth.pres)
+initMsg:SimCmdMsg
+run:boolean
+ep:EarthPanel
+PresM(SimBuffer,SimSettings,SimCmd
+pause():void
+resume():void
+run():void
+listen():void
+iterate():void
+start():void
+stop():void
PresS
(edu.gatech.heatedearth.pres)
+ep:EarthPanel
+PresS(SimBuffer,SimSettings)
+PresS()
+iterate():void
+pause():void
+run():void
+listen():void
+iterate():void
+start():void
+stop():void
+resume():void
+restart():void
SimM
(edu.gatech.heatedearth.sim)
+simulation:SimulatedEarth
-displayRate:int
-gridSize:int
-timeStep:int
+SimM(SimBuffer,SimSettings,SimCmdF
+UpdatePresentation():void
+iterate():void
+pause():void
+resume():void
+run():void
SimS
(edu.gatech.heatedearth.sim)
+displayRate:int
+gridSize:int
+timeStep:int
+stop:boolean
+simulation:SimulatedEarth
+SimS(SimBuffer,SimSettings)
+iterate():void
+pause():void
+resume():void
+run():void
+listen():void
+refresh():void
+start():void
+init():void
Simulation
(edu.gatech.heatedearth.sim)
+gridSpacing:int
+simTimeStep:int
Simulation()
iterateSimulation():SimState
+getBuffer:SimBuffer
+setBuffer(SimBuffer):void
+start():void
SimulatedEarth
(edu.gatech.heatedearth.sim)
+widthInDegrees:int
+heightInDegrees:int
+circumfrenceOfEarth:int
+surfaceAreaOfEarth:double
+initialCellTemperature:int
+degreesRotatedPerMinute:double
+heatPerInteration:double
+gridRows:int
+gridColumns:int
+proportionOfEquator:int
+earth:EarthCell[][]
+time:int
+rotation:double
+maxDelta:double
+SimulatedEarth(int)
+getCurrentSimulationState():SimState
+simulateIteration(int):void
+CoolEarth(double):void
+rotateGrid(double):void
+diffuseHeatFromNeighbors():double
+heatGridFromSun():void
+degreesToRotate():void
+CreateEarthCell(int,int):EarthCell
+CalculateLongitude(int):double
+CalculateLatitude(int):double
+CalculateAngleForRadiation(int,int,double):double
+GetWestNeighbor((int,int):EarthCell
+GetEastNeighbor((int,int):EarthCell
+GetNorthNeighbor((int,int):EarthCell
+GetSouthNeighbor((int,int):EarthCell
-demoFrame
0..1
-factory
0..1
-sf
0..1 simSettings
0..1
- simSettings
- simSettings
+simMsg
+initMsg
+presMsg
0..1 simSettings
0..1 control Panel
- simSettings
-factory
#buffer
0..1
#buffer
#pres
0..1
initMsg
#initReceiver
simBuffer
simReceiver
0..*
presMsg
pres
-simulation
1
settings

Sequence Diagram


The User supplies beginning parameters in on the command line which is parsed by the ARGVParser to generate a SimSettings object.
Main creates the User Control, using the simSetting object to generate the original state of the gui parameters.
The user initializes the simulation, and the simSettings object is passed to the SimulationFactory, to create the SimBuffer. The SimBuffer is passed to the Simulation, and Presentation in their constructor methods
Main/Demo
EarthPanel
EarthPanel:ep
SimSettingsargv)
SimSettings:settings
LoadEarthPanel(ep)
UserControls(settings)
UserControls
SimM
SimulationFactory
SimmCmdMsg
SimulationFactory
SimCmdMsg:initMsg
SimCmdReceivable
SimS
start
start()
buildSimulation(settings, initMsg)
addMsg(start)
buildSimulation():void
SimulationCtl
SimS(SimBuffer,SimSettings)
getBuffer()SimBuffer
invoke
resume():void
listen():void
return
iterate():void

Architecture

The *SimulatedEarth* class is a component with the responsibility of performing all calculations needed in order to drive the simulation of the Earth. On initialization, at which point it is provided with the size of the grid, the *SimulatedEarth* calculates information about all *EarthCells*, including their sizes and position relative to the sun. Its main function is to iterate the simulation by a unit of time provided, and calculate the effects of the sun's heat at that point in time. In the process of updating the temperatures of all cells the *SimulatedEarth* applies heat from the sun, spreads heat among *EarthCell* neighbors, and has the earth cool the same amount of heat that it gained in the heating process. The *SimulatedEarth* can also be prompted to return a *SimState* that represents the current state of the simulation.

An *EarthCell* component represents information about one section of the simulation of the heated Earth. It contains information including the side lengths of the cell, the area of the cell, the temperature of the cell, and the angle at which it is currently radiated. The temperature and angle of radiation can be updated, while the other values can only be read. The *EarthCell* is only used by the *SimulatedEarth*.

The *ControlPanel* component manages all information dealing with the user interface of the Heated Earth Simulation. The *ControlPanel* maintains a reference to the *Demo* which creates it, the current *SimSettings* of the simulation, as well as the *SimulationFactory* which instantiates the presenter and simulator used to drive the program. The *ControlPanel* also contains many different user interface controls from the Java swing library.

The *Demo* component is the entry point to the program and processes any command line arguments and instantiates the user interface. The *Demo* object contains a *SimulationFactory*, initializes an instance of the *EarthPlate*, initializes an instance of a *SimCmdReceivable* object,

Connectors:

The *SimBuffer* acts as a connector between the presentation and simulation. Objects can be pushed onto the *SimBuffer* and then retrieved. It is initialized with a set amount of space which cannot change.

The *SimM* controls the Heated Earth Simulation when the simulation is given the initiative, acting as a connector between the simulation and the presentation. The *SimM* maintains a reference to the *SimBuffer* in order to transmit data, and an object that implements the *SimCmdReceivable* that acts as a presenter. The *SimM* is responsible for requesting that the presenter update when it supplies new information, in the *SimSettings* format, via the *SimBuffer*. The *SimM* instantiates and maintains a reference to a *SimulatedEarth,* and controls it’s use.

The *SimS* component controls the Heated Earth Simulation when the simulation does not have the initiative, acting as a connector between the simulation and the presentation. The *SimS* maintains a reference to the *SimBuffer* in order to transmit data, and contains the current *SimSettings*. It also contains a *CmdListenable* object that allows the *SimS* to receive commands from the object with initiative. The *SimS* instantiates and maintains a reference to a *SimulatedEarth,* and controls its use.

The *PresM* component controls the presentation of the Heated Earth Simulation when the presentation is controlling the initiative of the program, acting as a connector between the simulation and the presentation. The *PresM* maintains a reference to the *EarthPanel* display in or to manage updates, the *SimBuffer* in order to receive information calculated by the simulation, a *SimCmdReceivable* object which it orders to calculate simulations, a *SimCmdMsg* in order to receive commands from the user interface, and a *SimSettings* object that maintains the current state of the simulation.

The *PresS* component controls the presentation of the Heated Earth Simulation when the presentation is not controlling the initiative of the program, acting as a connector between the simulation and the presentation. The *PresS* maintains a reference to the *EarthPanel* display in order to manage updates, the *SimBuffer* to receive information calculated by the simulation, a *CmdListenable* object with which it receives instructions from the object with the initiative, and a *SimSettings* object that maintains the current state of the simulation.

Depending on the configuration options picked by the user, the Heated Earth Simulation will use the *SimulationFactory* to either instantiate a *PresM* and *SimS* pair, a *SimM* and *PresS* pair, or a *SimS* and *PresS* pair. These then communicate with each other on the starting and stopping of the calculation, while passing data back and forth using the *SimBuffer*.

Low Level Design Considerations

The focus of many low level design considerations was the pursuit of easily readable and maintainable code. In the actual simulation algorithm an object was used to represent each cell on the simulated Earth, despite the fact that using multiple two dimensional arrays would be more efficient. We decided that using an object would help others to understand the algorithm more easily, and that this was more important than the speed penalty to the calculation.

NonFunctional Requirements:

1. The program takes no longer than a minute to stabilize.
    a. Design decisions: The grid spacing must be at least one degree..
    b. Justification: Grids with spacing of less than one degree cause exponentially larger time periods to stabilize. While this limits the extent of simulations made using the program, it also prevents users from extended (or unending) wait times.
2. The UI controls should allow for the easy modification of parameter values. This includes changing the initiative, changing the grid spacing or the threading of different operations, among others.
    a. Design decision: The UI should include all options that users can specify from the command line, so that they can simply modify these attributes and run the program again.

    b. Justification: Restarting the program is a high cost operation. Any options that we can provide that can prevent that, even if they clutter the interface, should be used.

3. Running simulations should always happen at the user's discretion in order to prevent accidental waiting.
    a. Design decision: A button is supplied that allows the user to decide when the right-hand side of the UI is re-rendered, instead running the simulation after each param change,
    b. Justification: While it might be interesting to see prompt changes, we view unneeded waits as prohibitive to that approach.

GUI Design

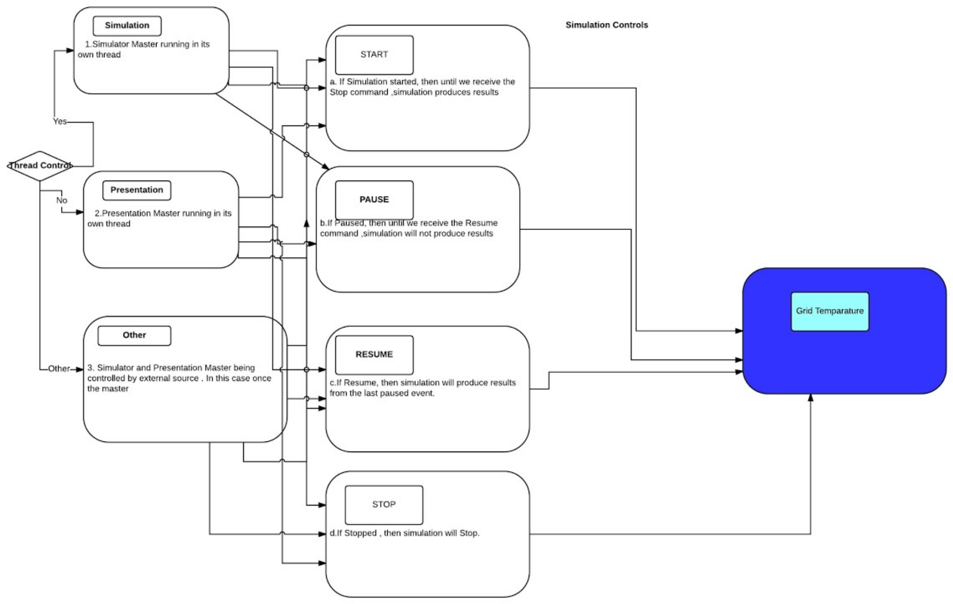
The main goal of the GUI for this project is to allow the user to quickly change the parameters of the simulation. The left hand navigation allows the user to control the various settings for each program. The results of the current simulation are displayed on the right side of the screen.

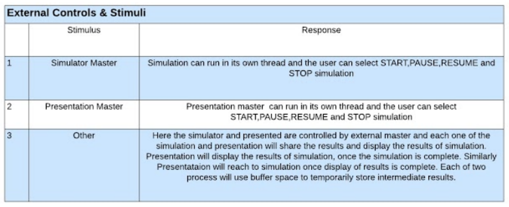
The controls pane provides four main sets of user interface controls. The upper portion of the UI allows the user to set the initiative, choosing between the Simulation as being the master, the Presentation being the Master, or a third party. Radio buttons are used because the decision on initiative is exclusive. The next portion of the interface, the Threading, consists of a pair of checkboxes in order for the user to indicate if either the Simulation or the Presentation should be on its own thread. We determined that checkboxes were the correct control because multiple options were valid.

Below the Thread portion of the interface the user is given four input boxes in order to select the Buffer Size, Grid Spacing, Time Step, and Display Rate. Because the user can enter an unbounded set of different values a textbox was determined to be the best control for these functions. On the bottom of the left side of the interface are four buttons which control the starting, pausing, resuming and stopping of the simulation.

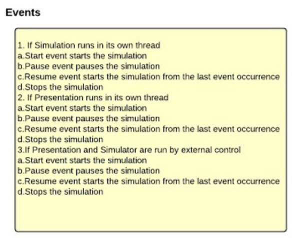
The rest of the interface window is comprised of the representation of the heated earth. An equirectangular projection of the earth is shown, color coded to show the current temperature. The temperatures are displayed in a grid that matches the grid used to drive the simulation, as specified by the user. As the Earth heats the temperatures turn from blue to red. Above the map the current location of the sun is shown. This configuration allows users to quickly access the current state of the Earth, and the position of the sun that creates it.

Statechart diagram:


Simulation Controls
Simulation
1.Simulator Master running in its own thread
Yes
Thread Control
No
Presentation
2.Presentation Master running in its own thread
Other
Other
3. Simulator and Presentation Master being controlled by external source . In this case once the master
START
a. If Simulation started, then until we receive the Stop command ,simulation produces results
PAUSE
b.If Paused, then until we receive the Resume command ,simulation will not produce results
RESUME
c.If Resume, then simulation will produce results from the last paused event.
STOP
d.If Stopped , then simulation will Stop.
Grid Temparature



External Controls & Stimuli		
	Stimulus	Response
1	Simulator Master	Simulation can run in its own thread and the user can select START,PAUSE,RESUME and STOP simulation
2	Presentation Master	Presentation master can run in its own thread and the user can select START,PAUSE,RESUME and STOP simulation
3	Other	Here the simulator and presented are controlled by external master and each one of the simulation and presentation will share the results and display the results of simulation. Presentation will display the results of simulation, once the simulation is complete. Similarly Presentataion will reach to simulation once display of results is complete. Each of two process will use buffer space to temporarily store intermediate results.



Events

1. If Simulation runs in its own thread
a.Start event starts the simulation
b.Pause event pauses the simulation
c.Resume event starts the simulation from the last event occurrence
d.Stops the simulation
2. If Presentation runs in its own thread
a.Start event starts the simulation
b.Pause event pauses the simulation
c.Resume event starts the simulation from the last event occurrence
d.Stops the simulation
3.If Presentation and Simulator are run by external control
a.Start event starts the simulation
b.Pause event pauses the simulation
c.Resume event starts the simulation from the last event occurrence
d.Stops the simulation

Code Reuse

When designing the Heated Earth Simulation the team attempted to use the lessons learned in the Heated Earth Design Study. Because of the complexity of the algorithm, and the number of moving pieces, we chose to use objects to represent the data in each Earth Cell. While there would have been some performance benefits to using an array of doubles, we felt that because we needed to keep track of multiple pieces of information about each cell this was not a workable solution. We could have engineered a solution that involved several grids but felt that the complexity of this arrangement would impede the development processes. Using this object model will also make the code written for this project much easier to use in future projects.

Reflection

While the design we created was adequate for the solution of the problem assigned, the design itself led to issues when troubleshooting. When troubleshooting issues with the project we spent a large amount of time dealing with issues brought about by the messaging bus, and issues with the refreshing of the user interface. This architecture was the most appropriate for the task at hand, yet was not implemented in such a way that allowed for easy maintenance. In hindsight, it may have been better to use a different architecture in order to improve the maintainability of the project.