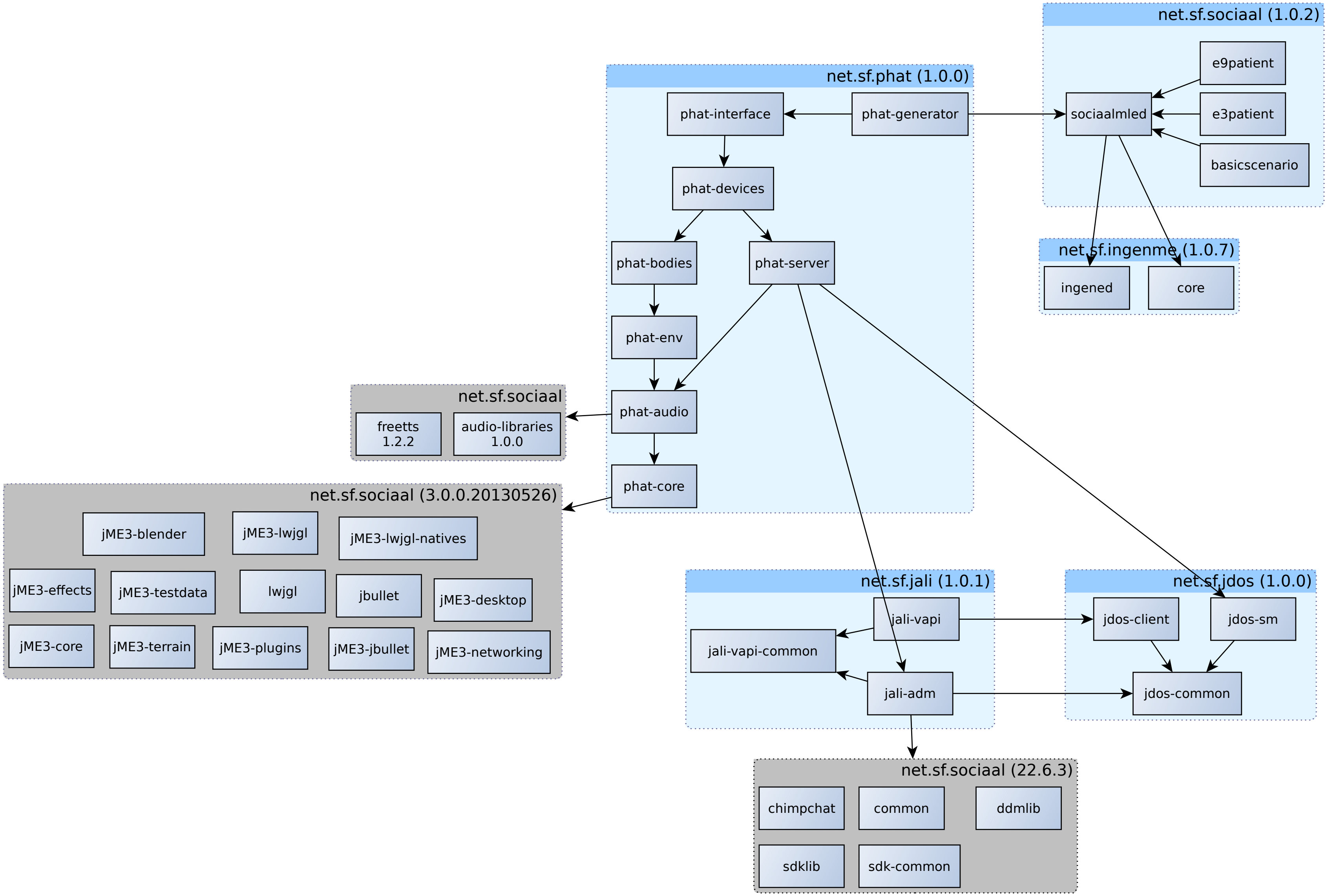
PHAT: Physical Human Activity Tester

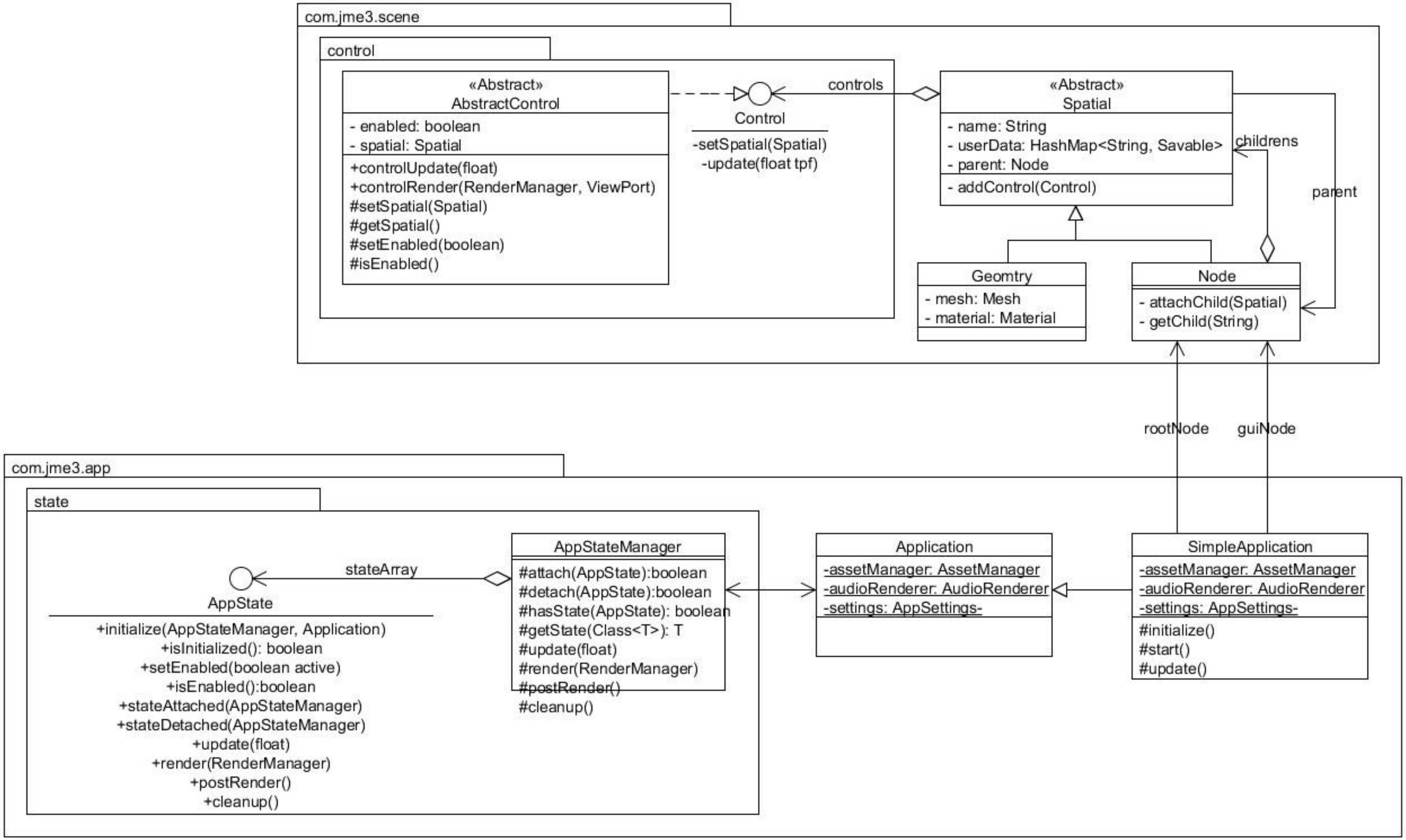
What features does it have and how are they built?

Pablo Campillo Sánchez

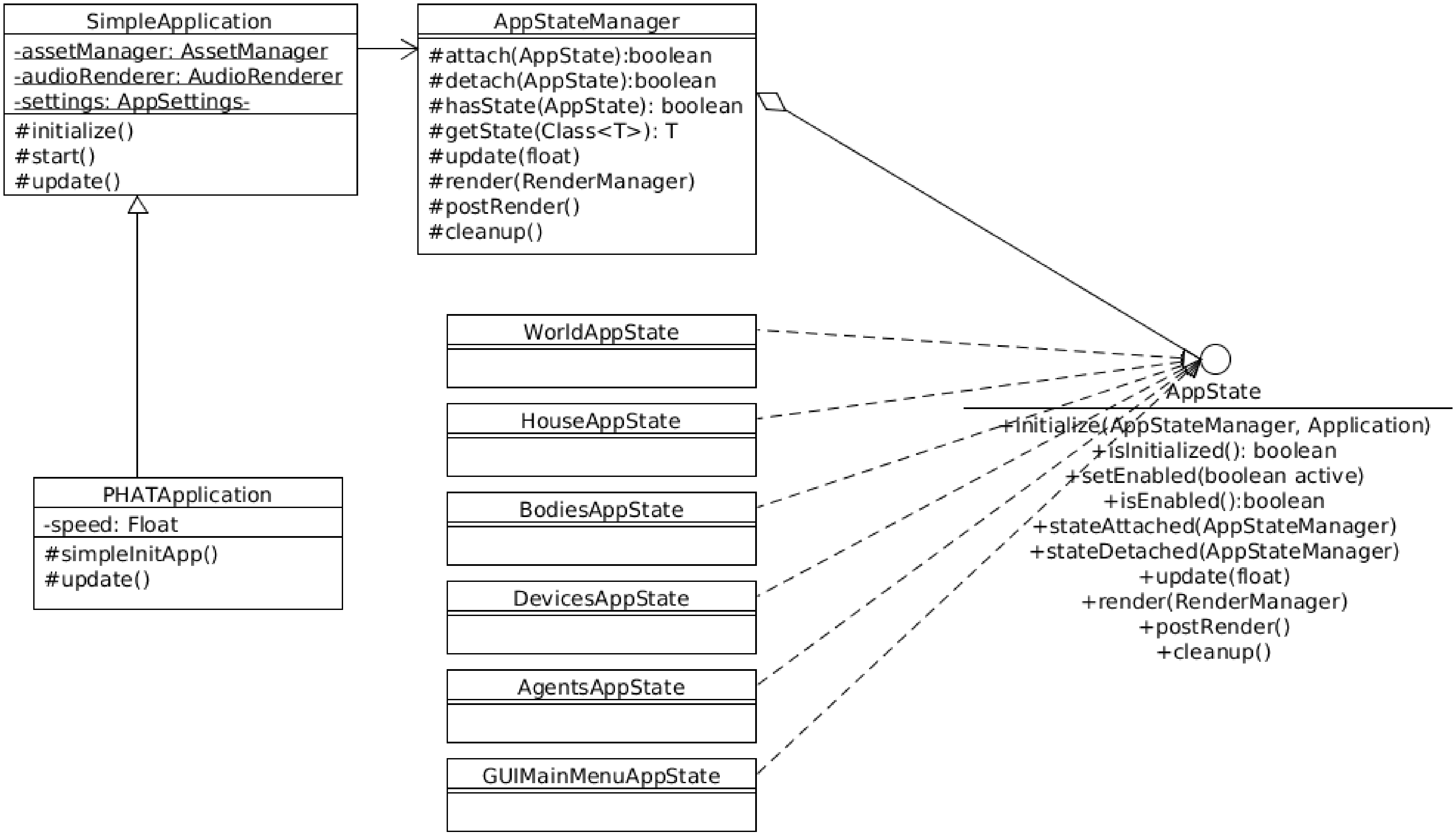
PHAT Artifacts



JME3 Control classes



Main PHAT Control classes



SimpleApplication

-assetManager: AssetManager
-audioRenderer: AudioRenderer
-settings: AppSettings-

#initialize()
#start()
#update()

AppStateManager

#attach(AppState):boolean
#detach(AppState):boolean
#hasState(AppState): boolean
#getState(Class<T>): T
#update(float)
#render(RenderManager)
#postRender()
#cleanup()

PHATApplication

-speed: Float

#simpleInitApp()
#update()

WorldAppState

HouseAppState

BodiesAppState

DevicesAppState

AgentsAppState

GUIMainMenuAppState

AppState

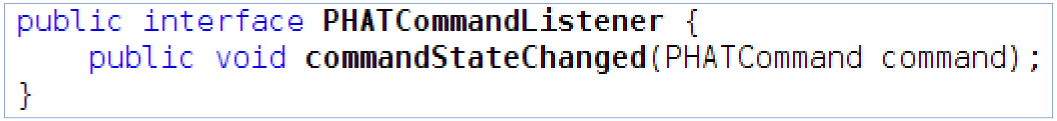
+initialize(AppStateManager, Application)
+isInitialized(): boolean
+setEnabled(boolean active)
+isEnabled():boolean
+stateAttached(AppStateManager)
+stateDetached(AppStateManager)
+update(float)
+render(RenderManager)
+postRender()
+cleanup()

Main Loop

- Initialization – Execute simpleInitApp() method once.
- Main Update Loop
  1. Input listeners respond to mouse clicks and keyboard presses – Input handling
  2. Update game state:
     1. Update overall simulation state – Execute Application States (WorldAppState, HouseAppState, BodiesAppState, etc.)
     2. PHATApplication code update – Execute update() method.
     3. Logical update of entities – Execute Custom Controls (Most of them are in bodies)
  3. Render audio and video
     1. Application States rendering.
     2. Scene rendering.
     3. User code rendering – Execute simpleRender() method.
  4. Repeat loop.
- **Quit** – If user requests **exit()**, execute **cleanup()** and **destroy()**.

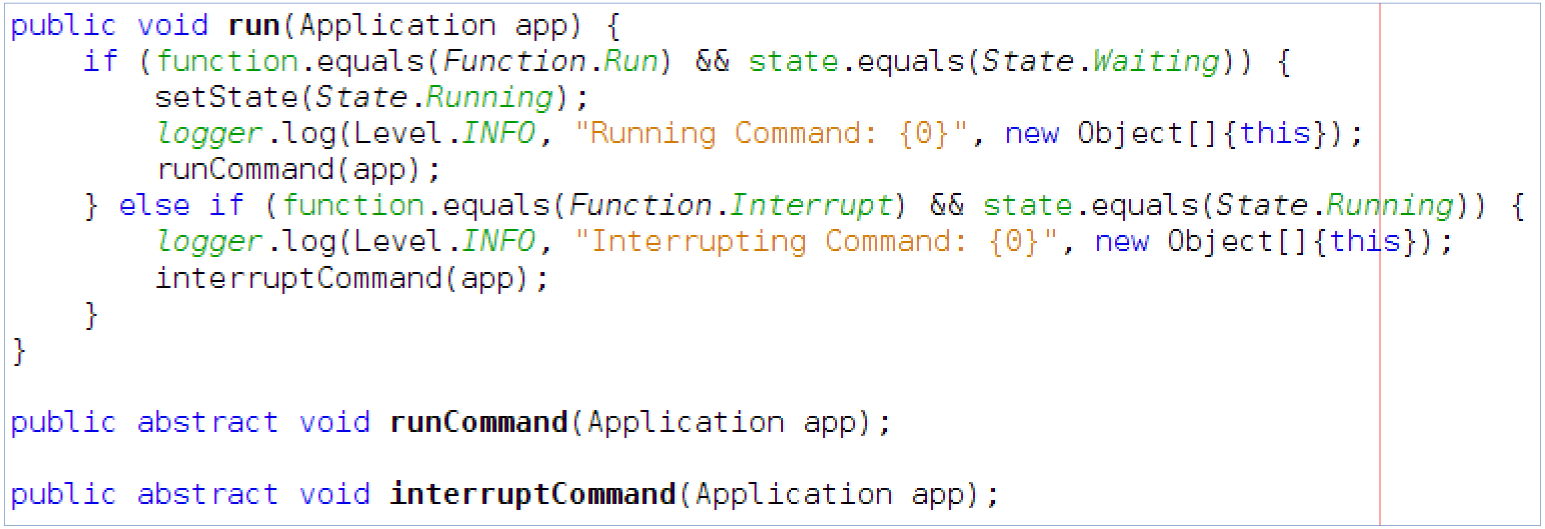
PHATCommands

- Functions: Run, Interrupt
- States: Waiting, Running, Interrupted, Success, Fail
- Notifies changes on its states to the Listener (PHATCommandListener)
- Asynchronous behaviour



```java
public interface PHATCommandListener {
    public void commandStateChanged(PHATCommand command);
}
```

PHATCommand functions:

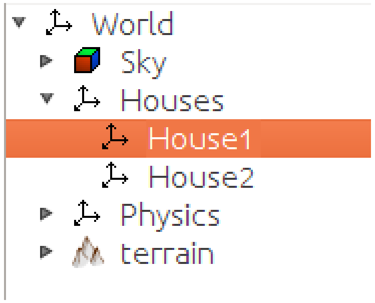


```java
public void run(Application app) {
    if (function.equals(Function.Run) && state.equals(State.Waiting)) {
        setState(State.Running);
        logger.log(Level.INFO, "Running Command: {0}", new Object[]{this});
        runCommand(app);
    } else if (function.equals(Function.Interrupt) && state.equals(State.Running)) {
        logger.log(Level.INFO, "Interrupting Command: {0}", new Object[]{this});
        interruptCommand(app);
    }
}

public abstract void runCommand(Application app);

public abstract void interruptCommand(Application app);
```

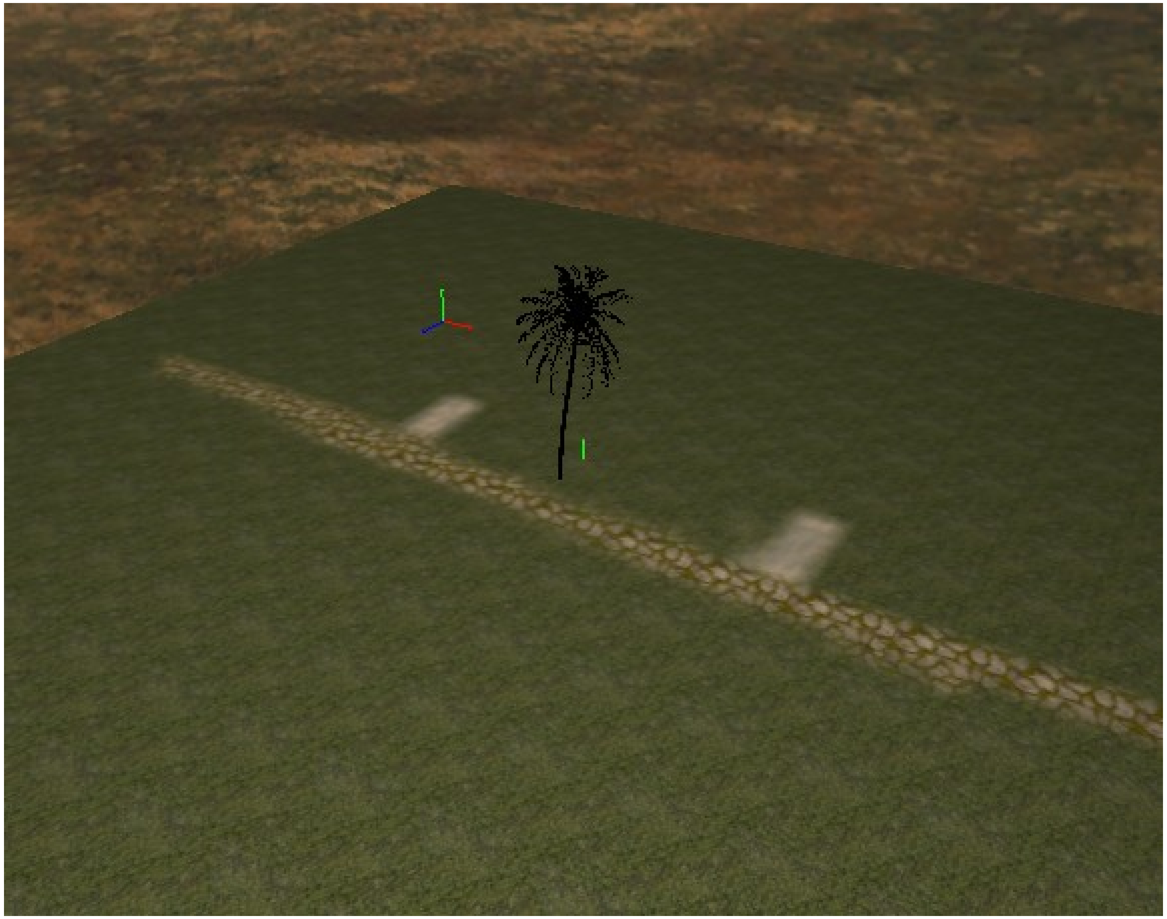
phat-env (1/6)

- WorldAppState:
  - Updates PHATCalendar (time and date)
  - Sky
  - Land and places where houses can be added
  - Init lights and simulates the sun
  - Weather no implemented

phat-env (2/6)



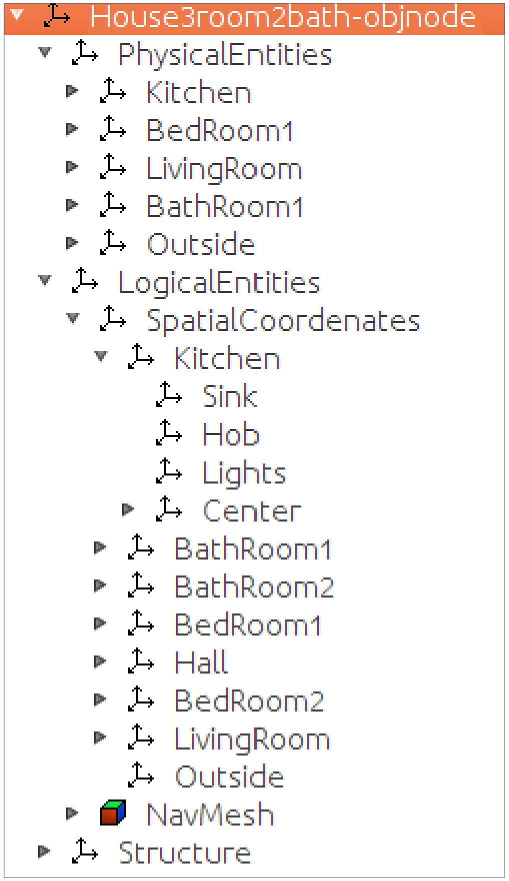
- World
  - Sky
  - Houses
    - House1
    - House2
  - Physics
  - terrain



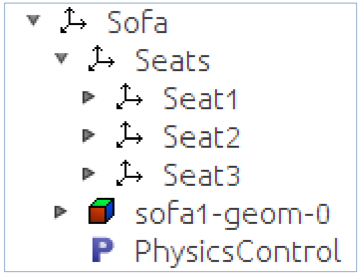
phat-env (3/6)

- HouseAppState:
  - NavMesh
  - Coordenates of areas and rooms.
  - Objects with meta information
  - Commands:
    - CreateHouseCommand(String id, HouseFactory.HouseType houseType)
    - DebugShowHouseNavMeshCommand(boolean enable)

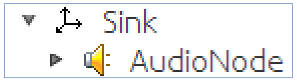
phat-env (4/6)



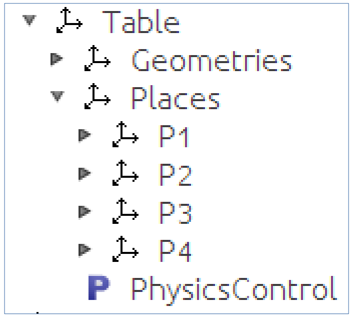
- House3room2bath-objnode
  - PhysicalEntities
    - Kitchen
    - BedRoom1
    - LivingRoom
    - BathRoom1
    - Outside
  - LogicalEntities
    - SpatialCoordenates
      - Kitchen
        - Sink
        - Hob
        - Lights
        - Center
      - BathRoom1
      - BathRoom2
      - BedRoom1
      - Hall
      - BedRoom2
      - LivingRoom
      - Outside
    - NavMesh
  - Structure



- Sofa
  - Seats
    - Seat1
    - Seat2
    - Seat3
  - sofa1-geom-0
  - PhysicsControl



- Sink
  - AudioNode



- Table
  - Geometries
  - Places
    - P1
    - P2
    - P3
    - P4
  - PhysicsControl

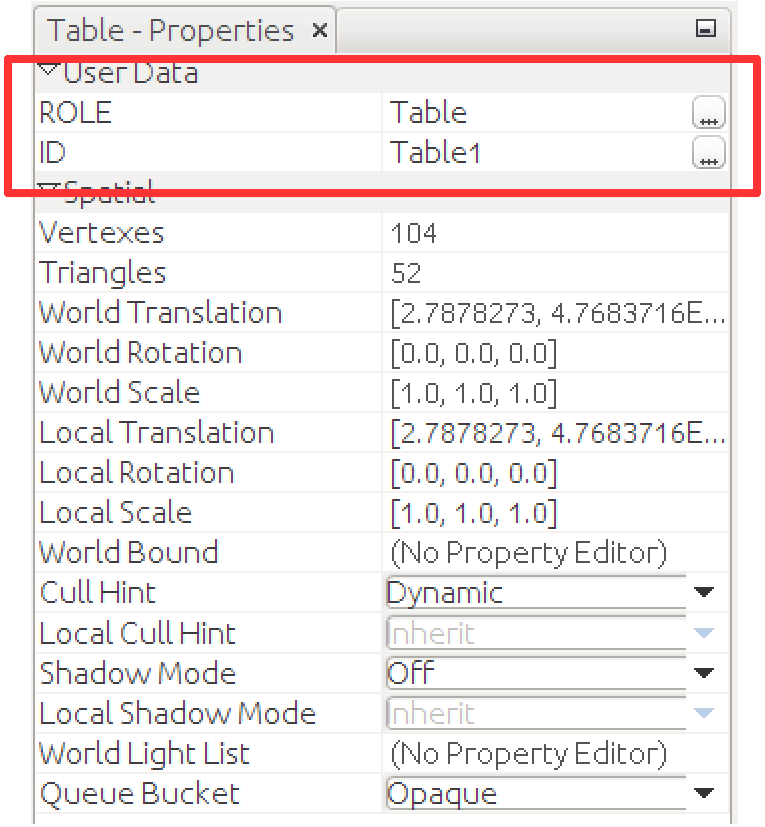


Table - Properties

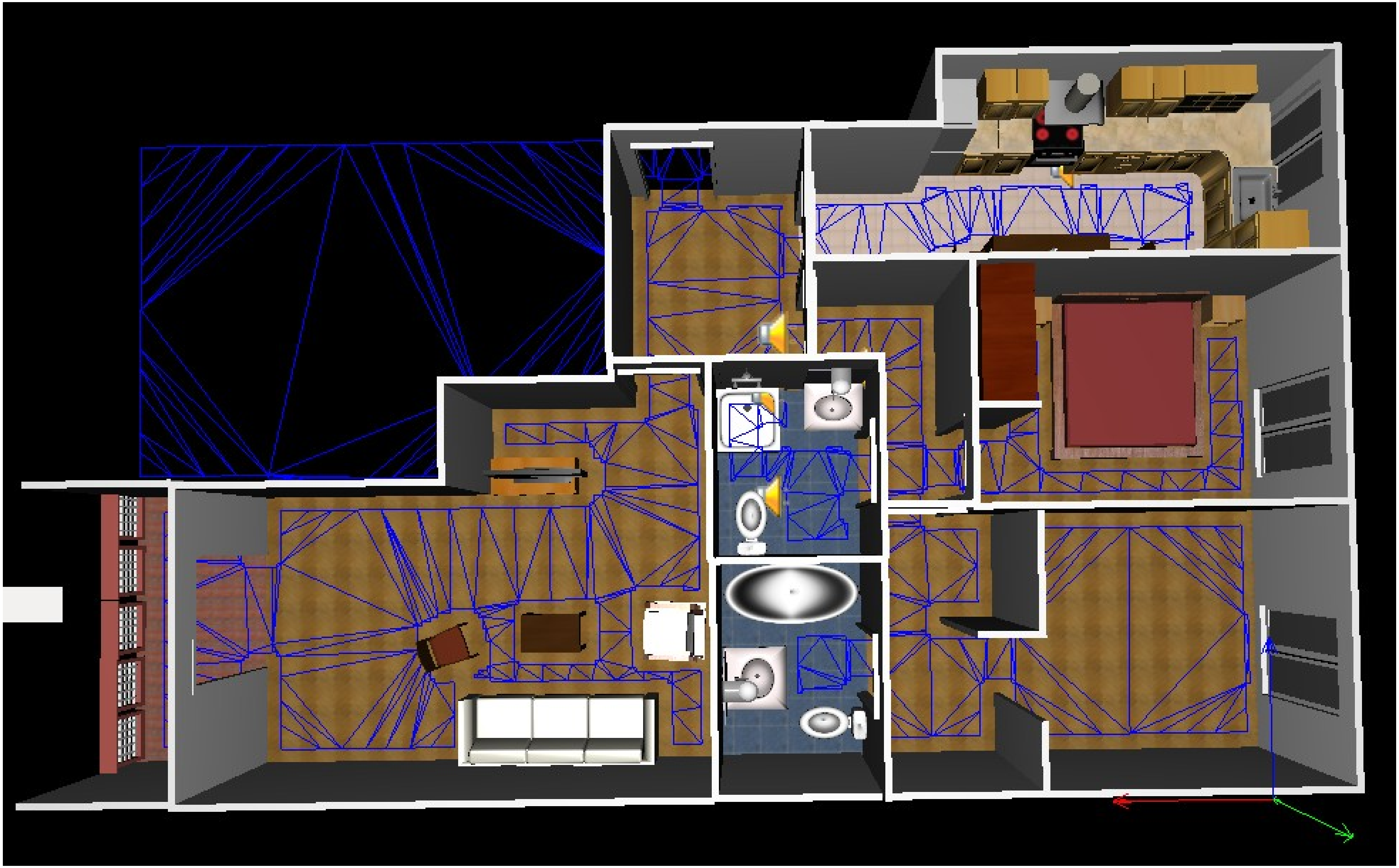
User Data

ROLE	Table
ID	Table1

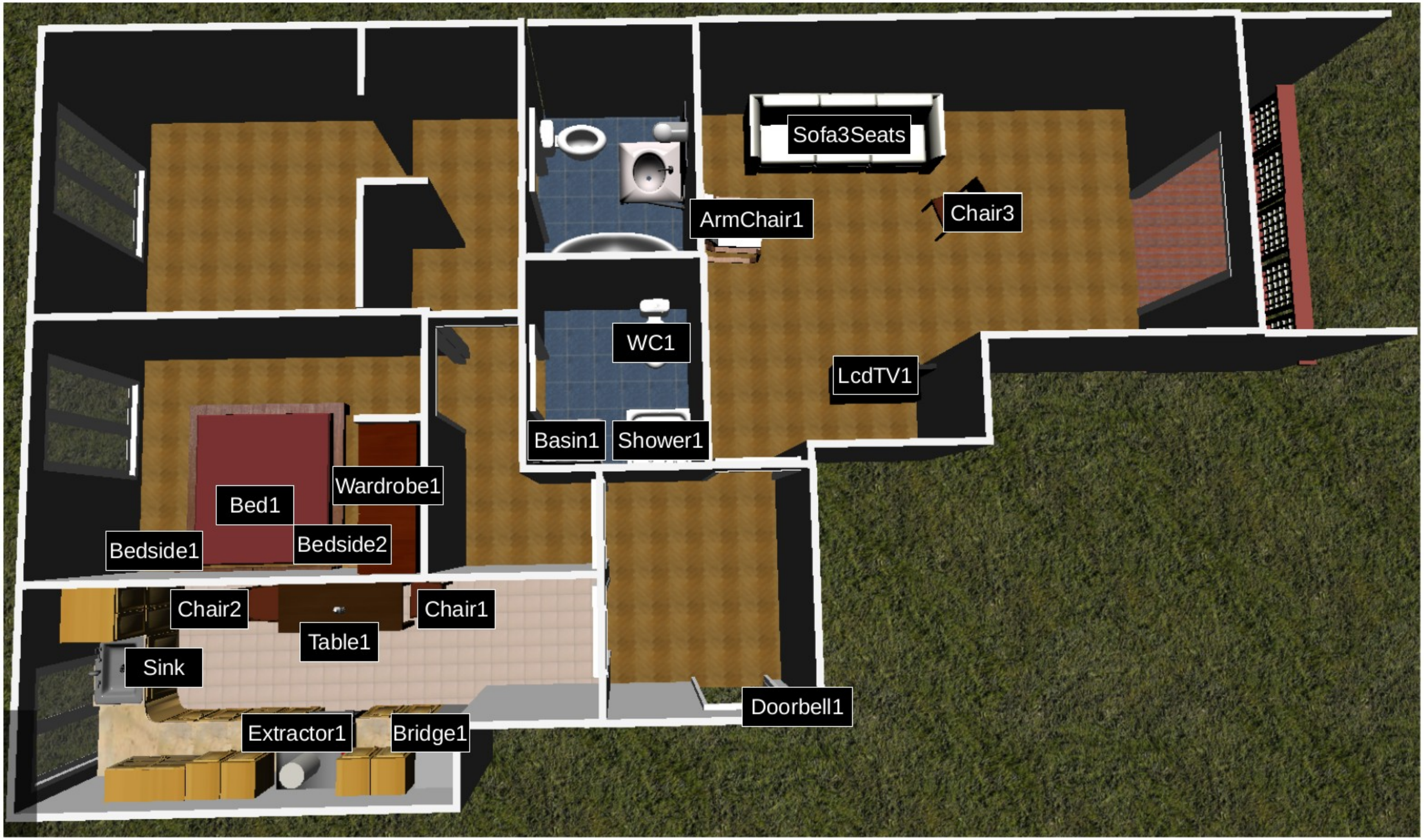
Spatial

Vertexes	104
Triangles	52
World Translation	[2.7878273, 4.7683716E...
World Rotation	[0.0, 0.0, 0.0]
World Scale	[1.0, 1.0, 1.0]
Local Translation	[2.7878273, 4.7683716E...
Local Rotation	[0.0, 0.0, 0.0]
Local Scale	[1.0, 1.0, 1.0]
World Bound	(No Property Editor)
Cull Hint	Dynamic
Local Cull Hint	Inherit
Shadow Mode	Off
Local Shadow Mode	Inherit
World Light List	(No Property Editor)
Queue Bucket	Opaque

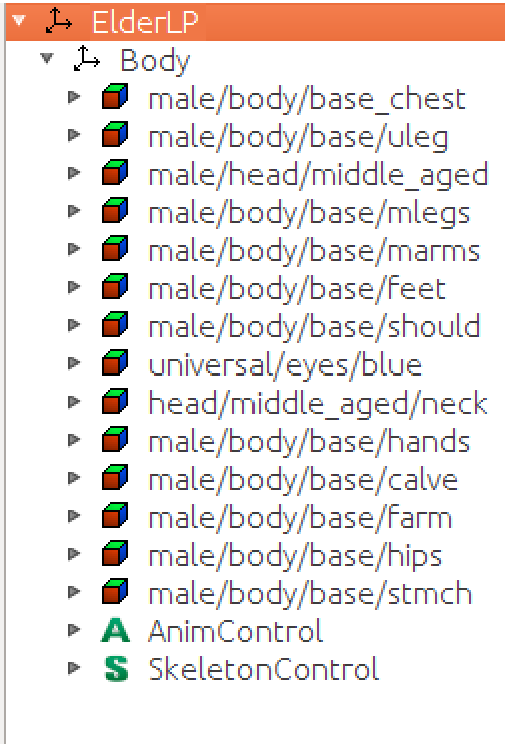
phat-env (5/6)



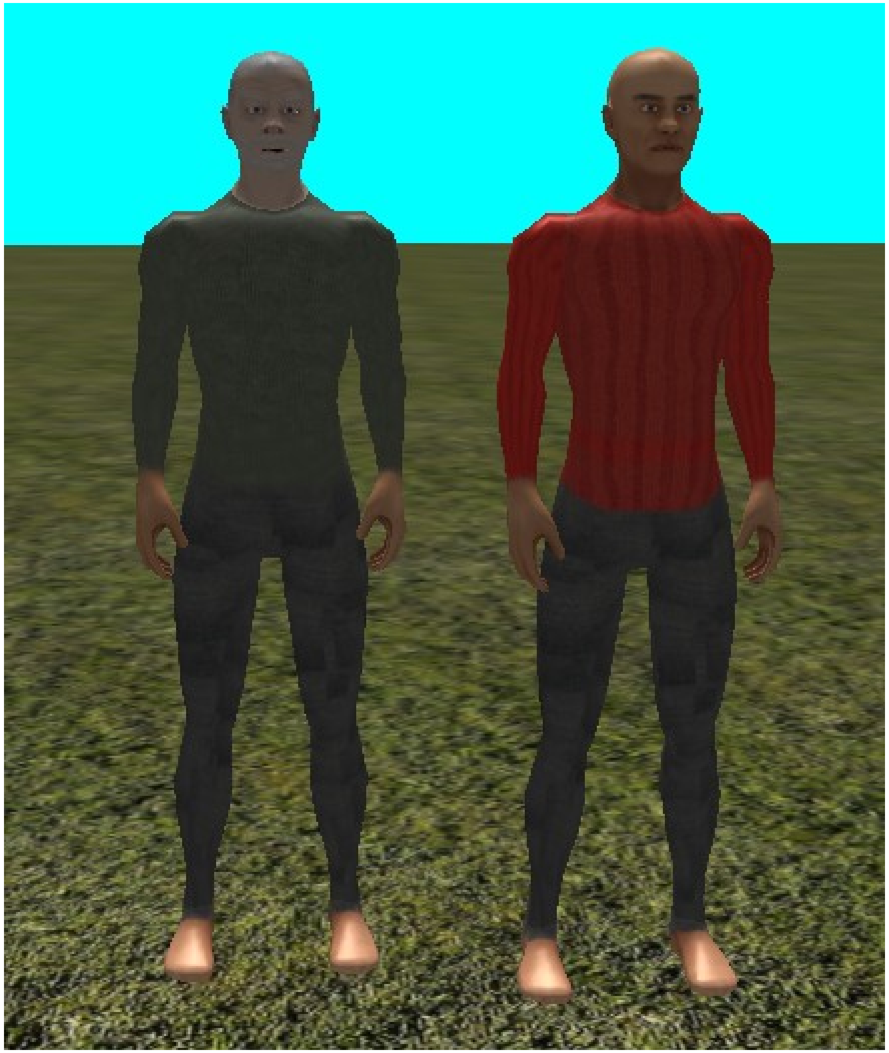
phat-env (6/6)


Sofa3Seats
ArmChair1
Chair3
WC1
LcdTV1
Basin1
Shower1
Wardrobe1
Bed1
Bedside1
Bedside2
Chair2
Chair1
Table1
Sink
Doorbell1
Extractor1
Bridge1

phat-bodies (1/6)

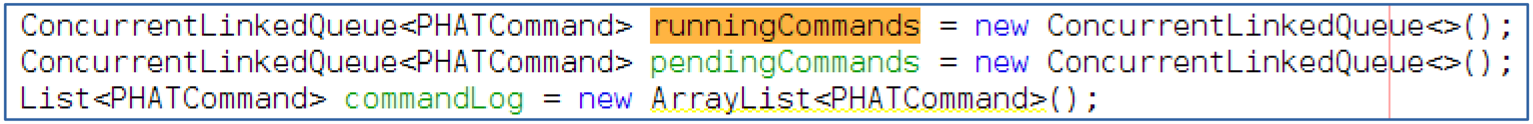


- ElderLP
  - Body
    - male/body/base_chest
    - male/body/base/uleg
    - male/head/middle_aged
    - male/body/base/mlegs
    - male/body/base/marms
    - male/body/base/feet
    - male/body/base/should
    - universal/eyes/blue
    - head/middle_aged/neck
    - male/body/base/hands
    - male/body/base/calve
    - male/body/base/farm
    - male/body/base/hips
    - male/body/base/stmch
    - AnimControl
    - SkeletonControl

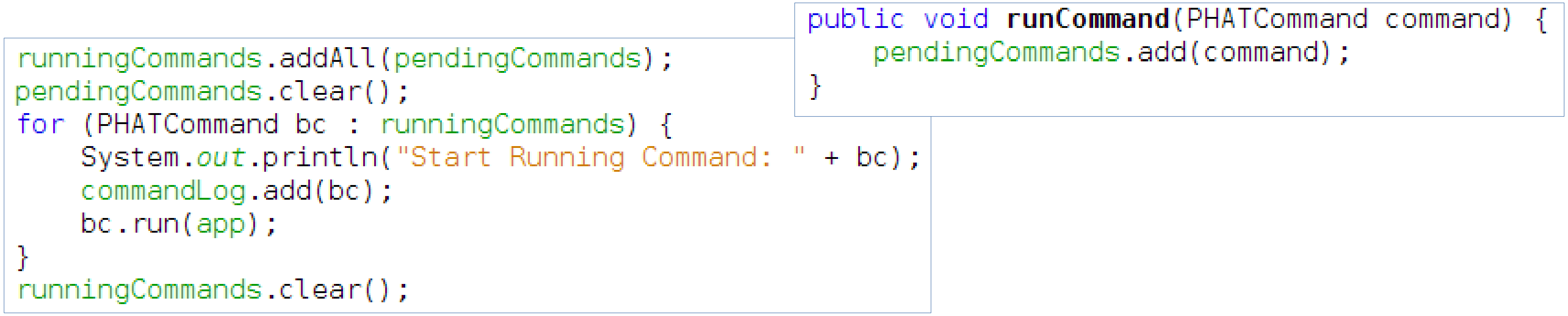


phat-bodies (2/6)

- BodiesAppState
  - Pending commands list both to be run and interrupted.
  - RunningCommands list that have started but don't ends.
  - A log of commands that has being runned → it is poor, it doesn't keep timestamp with state changes



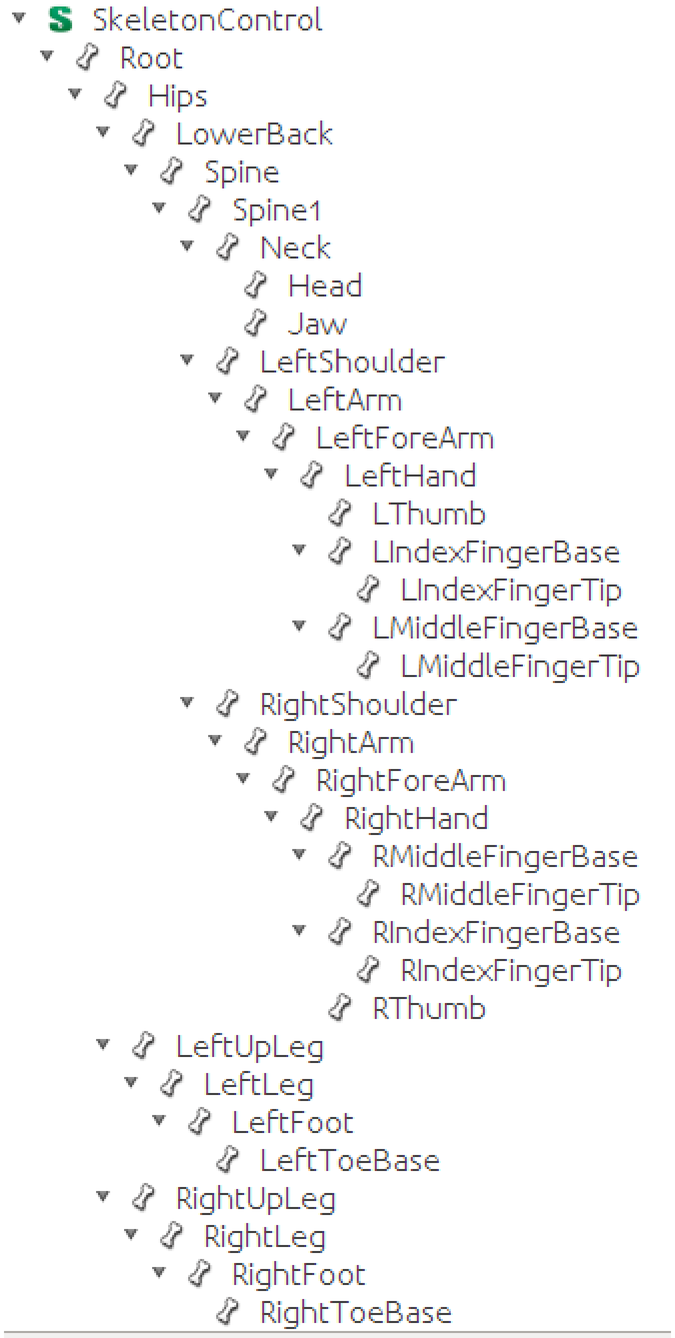
```
ConcurrentLinkedQueue<PHATCommand> runningCommands = new ConcurrentLinkedQueue<>();
ConcurrentLinkedQueue<PHATCommand> pendingCommands = new ConcurrentLinkedQueue<>();
List<PHATCommand> commandLog = new ArrayList<PHATCommand>();
```



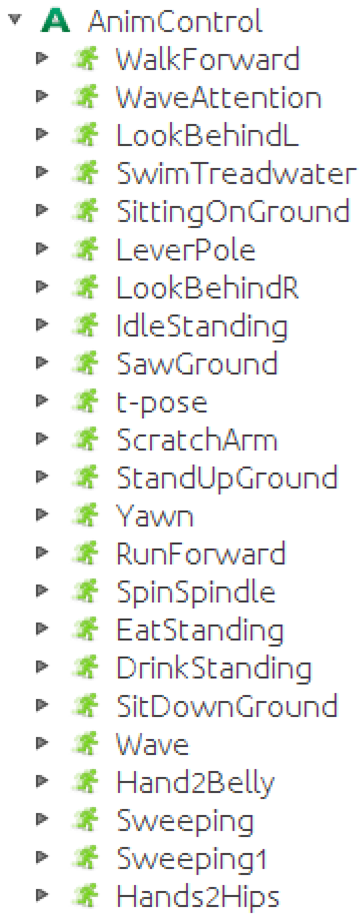
```
public void runCommand(PHATCommand command) {
    pendingCommands.add(command);
}
```

```
runningCommands.addAll(pendingCommands);
pendingCommands.clear();
for (PHATCommand bc : runningCommands) {
    System.out.println("Start Running Command: " + bc);
    commandLog.add(bc);
    bc.run(app);
}
runningCommands.clear();
```

Phat-bodies (3/6)



- SkeletonControl
  - Root
    - Hips
      - LowerBack
        - Spine
          - Spine1
            - Neck
              - Head
              - Jaw
            - LeftShoulder
              - LeftArm
                - LeftForeArm
                  - LeftHand
                    - LThumb
                    - LIndexFingerBase
                      - LIndexFingerTip
                    - LMiddleFingerBase
                      - LMiddleFingerTip
            - RightShoulder
              - RightArm
                - RightForeArm
                  - RightHand
                    - RMiddleFingerBase
                      - RMiddleFingerTip
                    - RIndexFingerBase
                      - RIndexFingerTip
                    - RThumb
      - LeftUpLeg
        - LeftLeg
          - LeftFoot
            - LeftToeBase
      - RightUpLeg
        - RightLeg
          - RightFoot
            - RightToeBase



- AnimControl
  - WalkForward
  - WaveAttention
  - LookBehindL
  - SwimTreadwater
  - SittingOnGround
  - LeverPole
  - LookBehindR
  - IdleStanding
  - SawGround
  - t-pose
  - ScratchArm
  - StandUpGround
  - Yawn
  - RunForward
  - SpinSpindle
  - EatStanding
  - DrinkStanding
  - SitDownGround
  - Wave
  - Hand2Belly
  - Sweeping
  - Sweeping1
  - Hands2Hips

phat-bodies (4/6)

- Commands
  - Displacements:
    - GoCloseToBodyCommand(String bodyId, String targetBodyId)
    - GoCloseToObjectCommand(String bodyId, String targetObjectId)
    - GoIntoBedCommand(String bodyId, String placeId)
    - GoToCommand(String bodyId, Lazy<Vector3f> destiny)
    - GoToSpaceCommand(String bodyId, String spaceId)
    - RemoveBodyFromSpaceCommand(String bodyId)
    - RotateTowardCommand(String bodyId, String entityId)
    - AlignWithCommand(String bodyId, String entityId)
    - SetBodyInCoordenatesCommand(String bodyId, Vector3f location)
    - SetBodyInHouseSpaceCommand(String bodyId, String houseId, String spaceId)
    - SetSpeedDisplacemenetCommand(String bodyId, float speed)
    - FallDownCommand(String bodyId)
    - TripOverCommand(String bodyId) → need to be reviewed

phat-bodies (5/6)

- Commands
  - Interactions:
    - OpenObjectCommand(String bodyId, String objectId)
    - CloseObjectCommand(String bodyId, String entityId)
    - PickUpCommand(String bodyId, String entityId, Hand hand)
    - SayASentenceBodyCommand(String bodyId, String message)
    - SitDownCommand(String bodyId, String placeId)
    - StandUpCommand(String bodyId)
  - Appearance:
    - SetBodyColorCommand(String bodyId, ColorRGBA color)
    - SetBodyHeightCommand(String bodyId, float height)
  - Debbuging:
    - AttachIconCommand(String bodyId, String imagePath, Boolean show)
    - BodyLabelCommand(String bodyId, Boolean show)
    - ShowLabelsOfVisibleObjectsCommand(String bodyId, boolean on)
    - SetCameraToBodyCommand(String bodyId)

phat-bodies (6/6)

- Commands
  - Gestures:
    - PlayBodyAnimationCommand(String bodyId, String animationName)
    - LookAtCommand(String bodyId, String targetId)
    - SetShortStepsCommand(String bodyId, Boolean on)
    - SetStoopedBodyCommand(String bodyId, Boolean on)
    - SetRigidArmCommand(String bodyId, Boolean on, Boolean left)
    - TremblingHandCommand(String bodyId, Boolean on, Boolean left)
    - TremblingHeadCommand(String bodyId, Boolean on)
  - Others:
    - SetPCListenerToBodyCommand(String bodyId)
    - WaitForCloseToBodyCommand(String bodyId, String targetBodyId)
    - CreateBodyTypeCommand(String bodyId, String urlResource)
    - RandomWalkingCommand(String bodyId, boolean enabled)

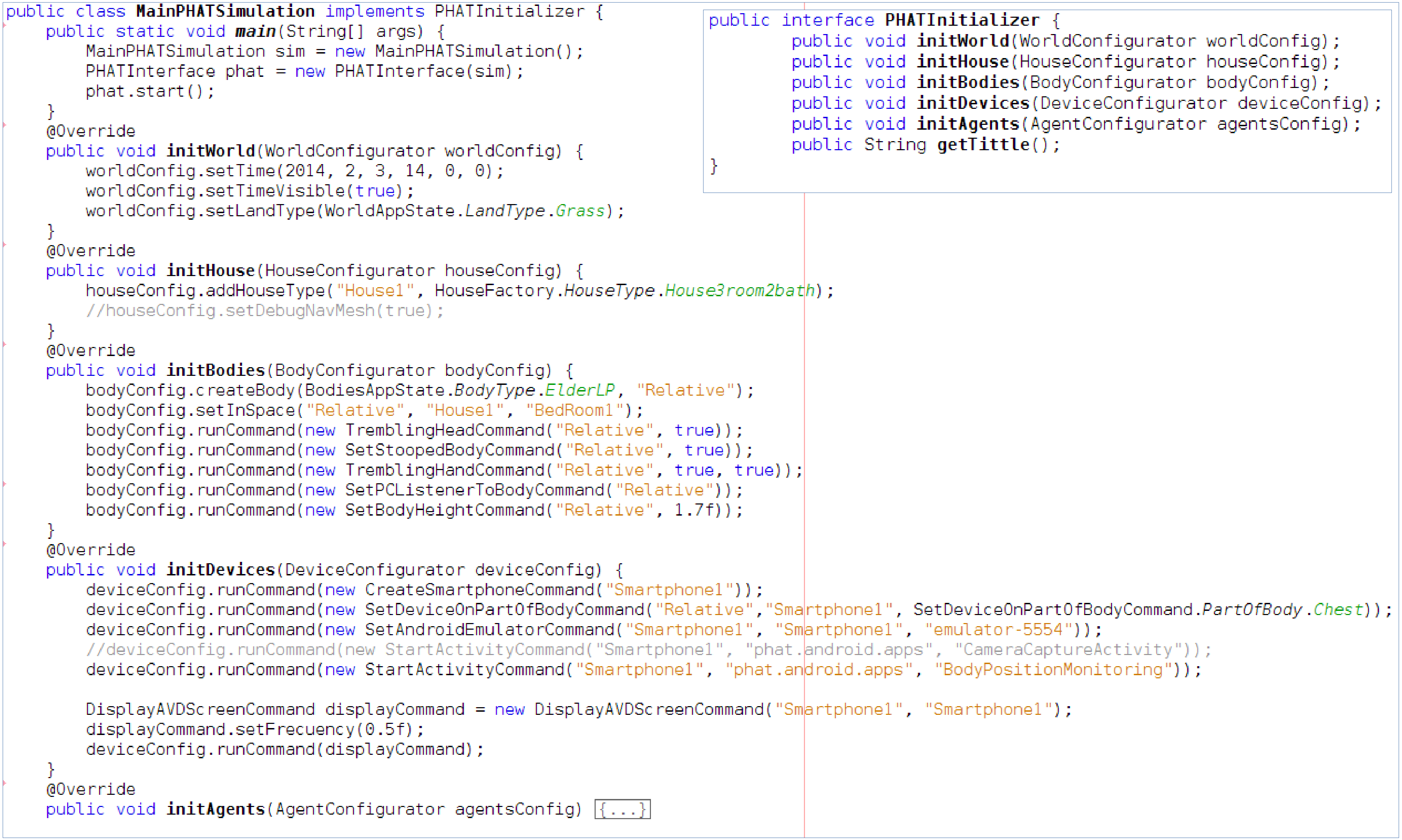
phat-devices

- There are not models of Android devices. They are created with a cube shape.
- DevicesAppState
- Commands:
  - CreateSmartphoneCommand(String smartphoneId)
  - DisplayAVDScreenCommand(String smartphoneId, String avdId)
  - InstallApkCommand(String smartphoneId, String apkFile)
  - PressOnScreen(String smartphoneId, int x, int y)
  - SetAndroidEmulatorCommand(String smartphoneId, String avdId, String serialEmulator)
  - SetDeviceInCoordenatesCommand(String deviceId, Vector3f location)
  - SetDeviceOnFurnitureCommand(String smartphoneId, String houseId, String furnitureId)
  - SetDeviceOnPartOfBodyCommand(String bodyId, String deviceId, PartOfBody partOfBody)
  - SetImageOnScreenCommand(String deviceId, String imagePath)
  - StartActivityCommand(String smartphoneId, String packageName, String activityName)
  - SwitchTVCommand(String tvId, boolean on) → deprecated

phat-interface

- Several responsibilities:
  - **AgentsAppState:** manage agents
  - **Agents:**
    - Automaton (HFSM)
    - PHATEventManager
    - DiseaseManager and Filters
  - **GUIMainMenuAppState**: graphical components to show FPS, take snapshot, pause and **forward** simulation.
  - **PHATInitializer**: facility to create and initializate the simulation.
    - Set a **seed** for repeteable experiments

Phat-interface - PHATInitializer



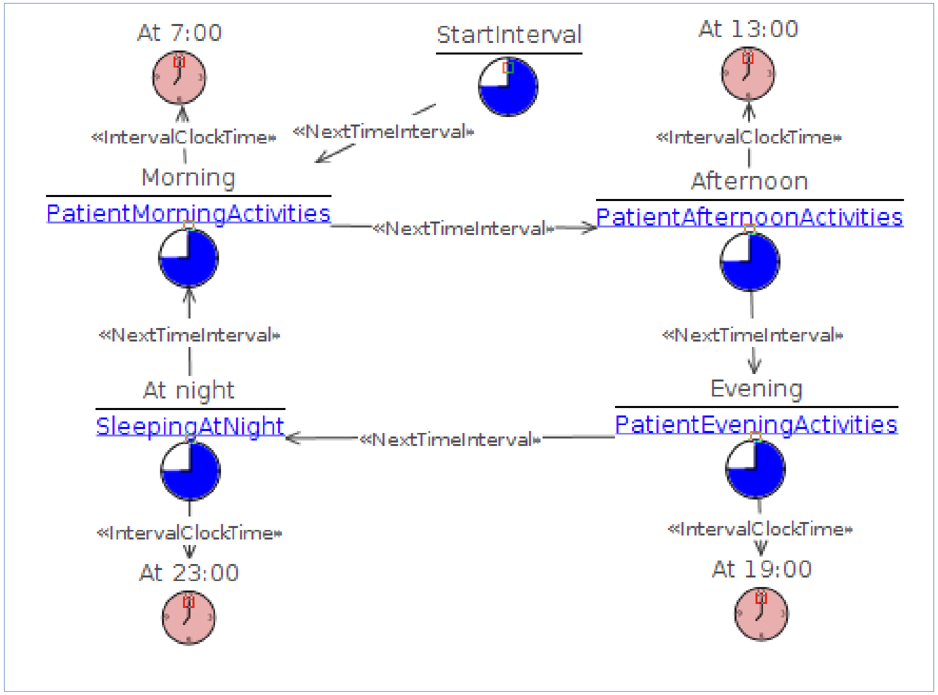
```java
public class MainPHATSimulation implements PHATInitializer {
    public static void main(String[] args) {
        MainPHATSimulation sim = new MainPHATSimulation();
        PHATInterface phat = new PHATInterface(sim);
        phat.start();
    }
    @Override
    public void initWorld(WorldConfigurator worldConfig) {
        worldConfig.setTime(2014, 2, 3, 14, 0, 0);
        worldConfig.setTimeVisible(true);
        worldConfig.setLandType(WorldAppState.LandType.Grass);
    }
    @Override
    public void initHouse(HouseConfigurator houseConfig) {
        houseConfig.addHouseType("House1", HouseFactory.HouseType.House3room2bath);
        //houseConfig.setDebugNavMesh(true);
    }
    @Override
    public void initBodies(BodyConfigurator bodyConfig) {
        bodyConfig.createBody(BodiesAppState.BodyType.ElderLP, "Relative");
        bodyConfig.setInSpace("Relative", "House1", "BedRoom1");
        bodyConfig.runCommand(new TremblingHeadCommand("Relative", true));
        bodyConfig.runCommand(new SetStoopedBodyCommand("Relative", true));
        bodyConfig.runCommand(new TremblingHandCommand("Relative", true, true));
        bodyConfig.runCommand(new SetPCListenerToBodyCommand("Relative"));
        bodyConfig.runCommand(new SetBodyHeightCommand("Relative", 1.7f));
    }
    @Override
    public void initDevices(DeviceConfigurator deviceConfig) {
        deviceConfig.runCommand(new CreateSmartphoneCommand("Smartphone1"));
        deviceConfig.runCommand(new SetDeviceOnPartOfBodyCommand("Relative","Smartphone1", SetDeviceOnPartOfBodyCommand.PartOfBody.Chest));
        deviceConfig.runCommand(new SetAndroidEmulatorCommand("Smartphone1", "Smartphone1", "emulator-5554"));
        //deviceConfig.runCommand(new StartActivityCommand("Smartphone1", "phat.android.apps", "CameraCaptureActivity"));
        deviceConfig.runCommand(new StartActivityCommand("Smartphone1", "phat.android.apps", "BodyPositionMonitoring"));

        DisplayAVDScreenCommand displayCommand = new DisplayAVDScreenCommand("Smartphone1", "Smartphone1");
        displayCommand.setFrecuency(0.5f);
        deviceConfig.runCommand(displayCommand);
    }
    @Override
    public void initAgents(AgentConfigurator agentsConfig) {...}
```

```java
public interface PHATInitializer {
        public void initWorld(WorldConfigurator worldConfig);
        public void initHouse(HouseConfigurator houseConfig);
        public void initBodies(BodyConfigurator bodyConfig);
        public void initDevices(DeviceConfigurator deviceConfig);
        public void initAgents(AgentConfigurator agentsConfig);
        public String getTittle();
}
```

SociAALML Behaviors

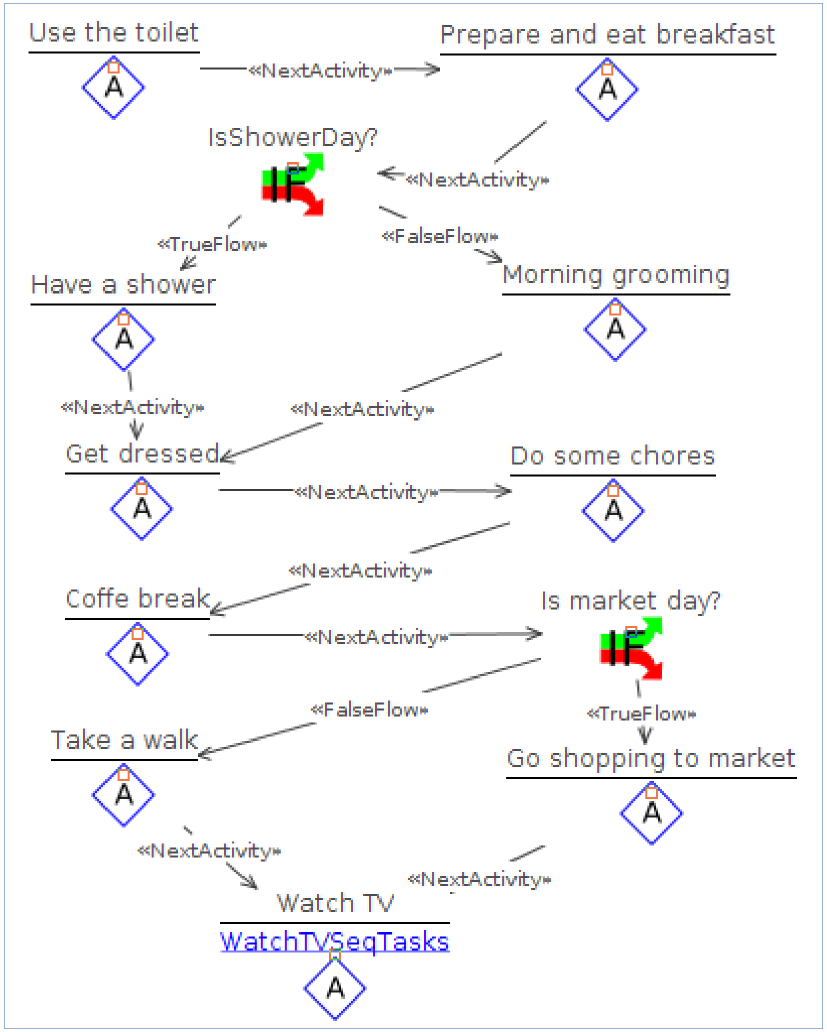
- ADLs are manage hierarchically:
  - Periods of time of the day
  - Activities
  - Tasks/Actions
- The purpose of these elements is to model the ADLs of a human in his/her house.
- **They assume that all activities and task will be carry out without problems.**
- These elements do not model human-human interaction, just human-environment (devices also).

SociAALML – Periods of time of the day



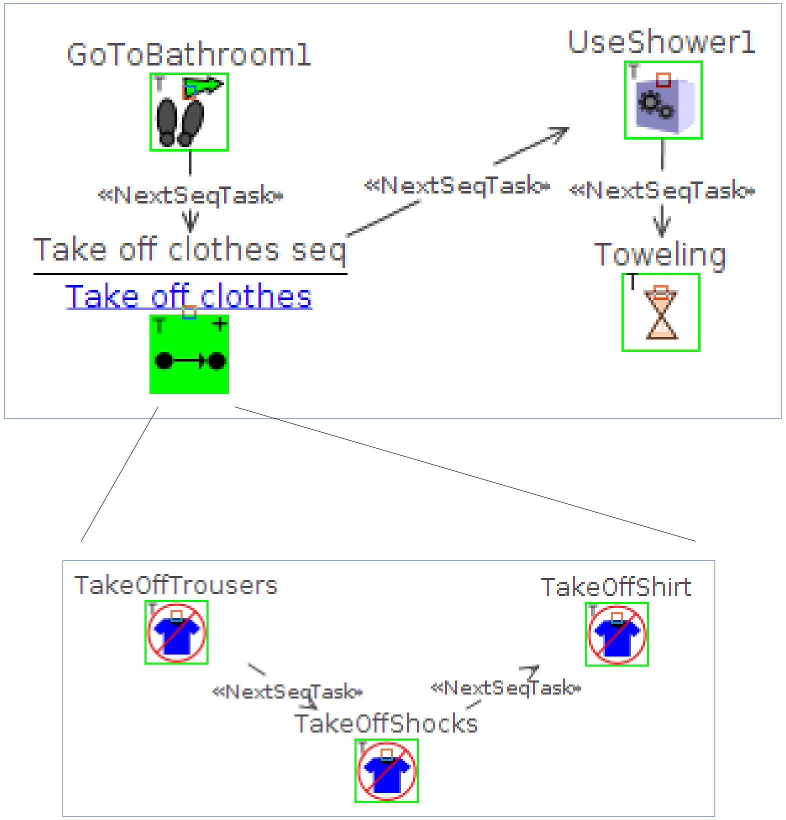
- ADLSpecDiagram: specifies a set of period of time that references to an AcivityDiagram.
- The clock indicates when the period starts.
- When it's time to start a new period it starts immediately and any task or activity in previous period is interrupted.
- What happen if all activities in the period have finished and It's no time to the next period? 2 options:
  - Define activity diagram such as it never ends.
  - Define an a default activity.

SociAALML – Activities



- ActivityDiagram: specifies a set of activities for a period of time.
- The activities can vary depending on conditions. They should be improved:
  - Probability
  - Level of a symptom (TODO)
  - Day of the week (TODO)
  - …
- When all task of an activity are finished then the next task starts.
- The last task has to be long time enough for the next period. Or modify the diagram to be able to add a default activity. Or it is possible to make a loop with two or more activities.

SociAALML – Tasks



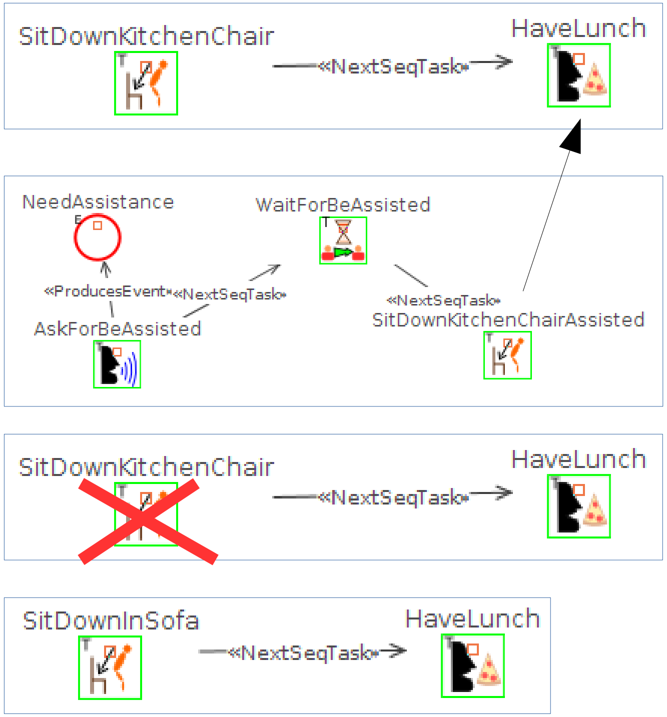
- Tasks can be defined using:
  - SequentialTaskDiagram: Task are performed in order.
  - RandomTaskDiagram: All tasks are performed in random order. (TODO)
- Both diagrams can contains tasks and elements that referenciate to a sequential or random task diagram.
- Activities just only be defined using SequentialTaskDiagram.
- Activities should start with a task to go to the place where it will be performed.

SociAALML – Filters

- Filters process tasks and modify them.
- They are activated depending on symptoms levels (NONE, LOW, MEDIUM, HIGH).
- Type of filters:
  - **Selector**: select what tasks will be sent to the next filter.
  - **Delay**: Changes the duration of a task or the speed.
  - **Unable**: The task is skipped, i.e., it is not performed and go on the next.
  - **Replace**: Remove the current task and add a new sequence of task.
  - **ChangeTarget**: Modifies a parameter of the task, e.g., change the chair where the patient will be seated. (TODO)
    - Ambiguous: Which parameter? They are not typified.

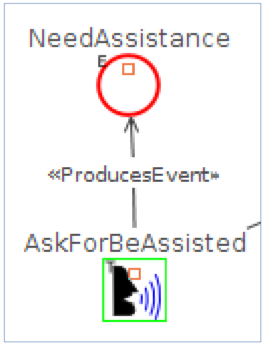
SociAALML – Filters

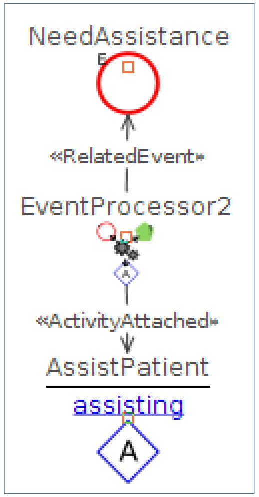
- **Look out with Unable, Replace and ChangeTarget filters:** A filter should be consistent with the task replaced to go on with the next task.
- For example, for the task sit down in a chair:
  - If a patient cannot perform the task the filter Replace can contain tasks to ask for help and so he/she is assisted and the task is performed.
  - Don't skip a task if it is important for the next task. Patient will eat standing!
  - Don't change the seat where he/she will be seated if it doesn't make sense, e.g. in the toilet.



SociAALML – Events

- **Events** can be attached to tasks in Sequential and Random Task Diagrams.
- Humans have a **InteractionProfile** that refers to an **InteractionDiagram** where mappings between events and activities are defined.
  - Needed condition to perform the activity can be added, but they are very limited yet.
- When a human is going to perform the activity of the event:
  - He/she suspends his/her current activity or task (or wait to be finished if it cannot be interrupted).
  - He resumes the interrupted activity (Resume an activity is complex, depending on what he/she were doing some task should be done before. For instance, to resume a sleeping task, the patient should go into bed first). It should be improved!
- It's supposed that there are several types of events depending on the medium they are perceived.
  - AudioEvent → has a volume and distance and it can be perceived just in the moment it was created. They are launched when the task is finished.
  - VisualEvent → It is not defined yet. The idea is that it was perceived only by direct vision. The event is launched at beginning of the task and it remains while the task is happening.
  - TouchEvent → It is not defined yet.


NeedAssistance
«ProducesEvent»
AskForBeAssisted


NeedAssistance
«RelatedEvent»
EventProcessor2
«ActivityAttached»
AssistPatient
assisting
A