Combination of Domain-Specific Modeling Languages

Practical Research

Rafael Ugaz
http://msdl.cs.mcgill.ca/people/rafael/

University of Antwerp

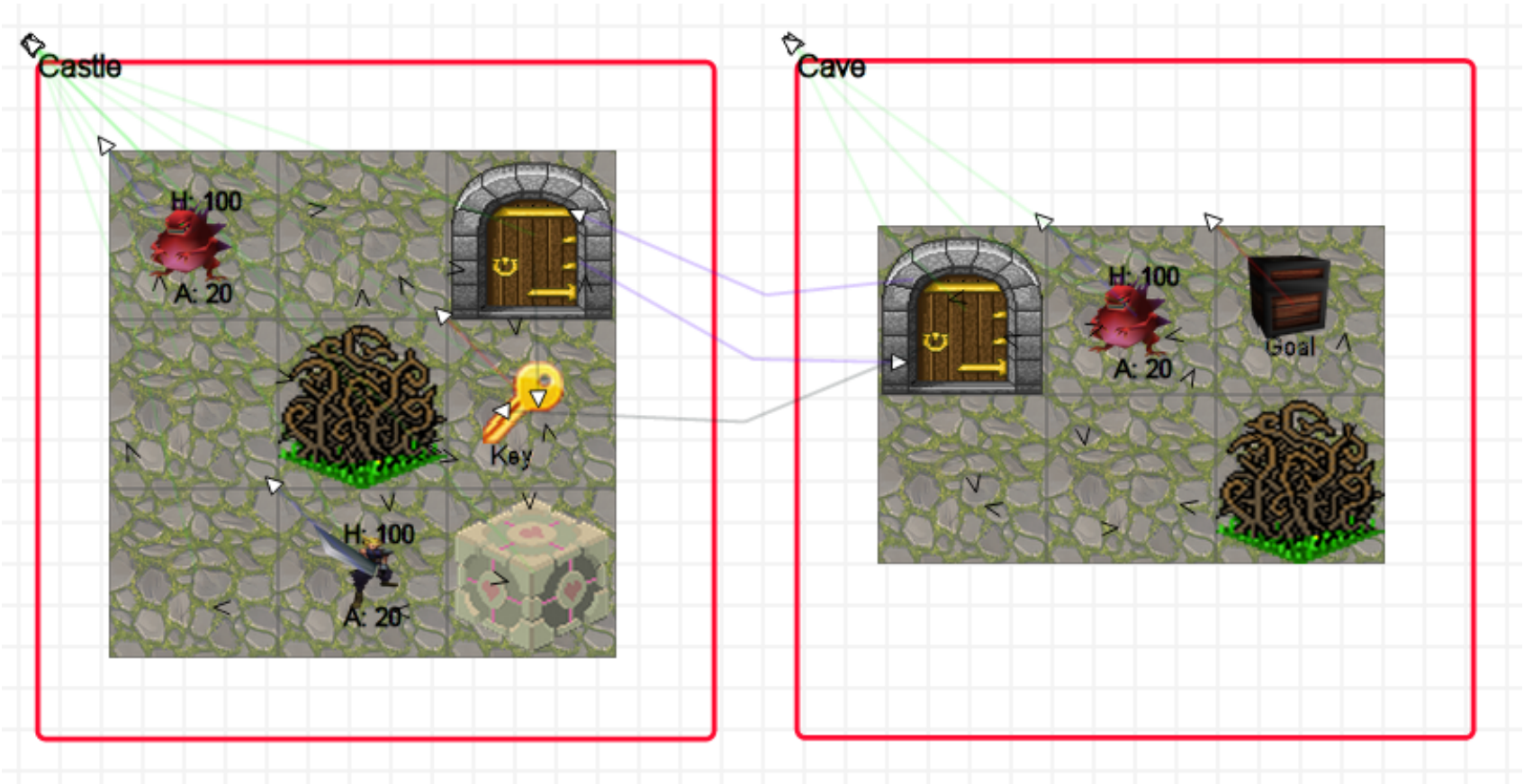
June 2014

Goal

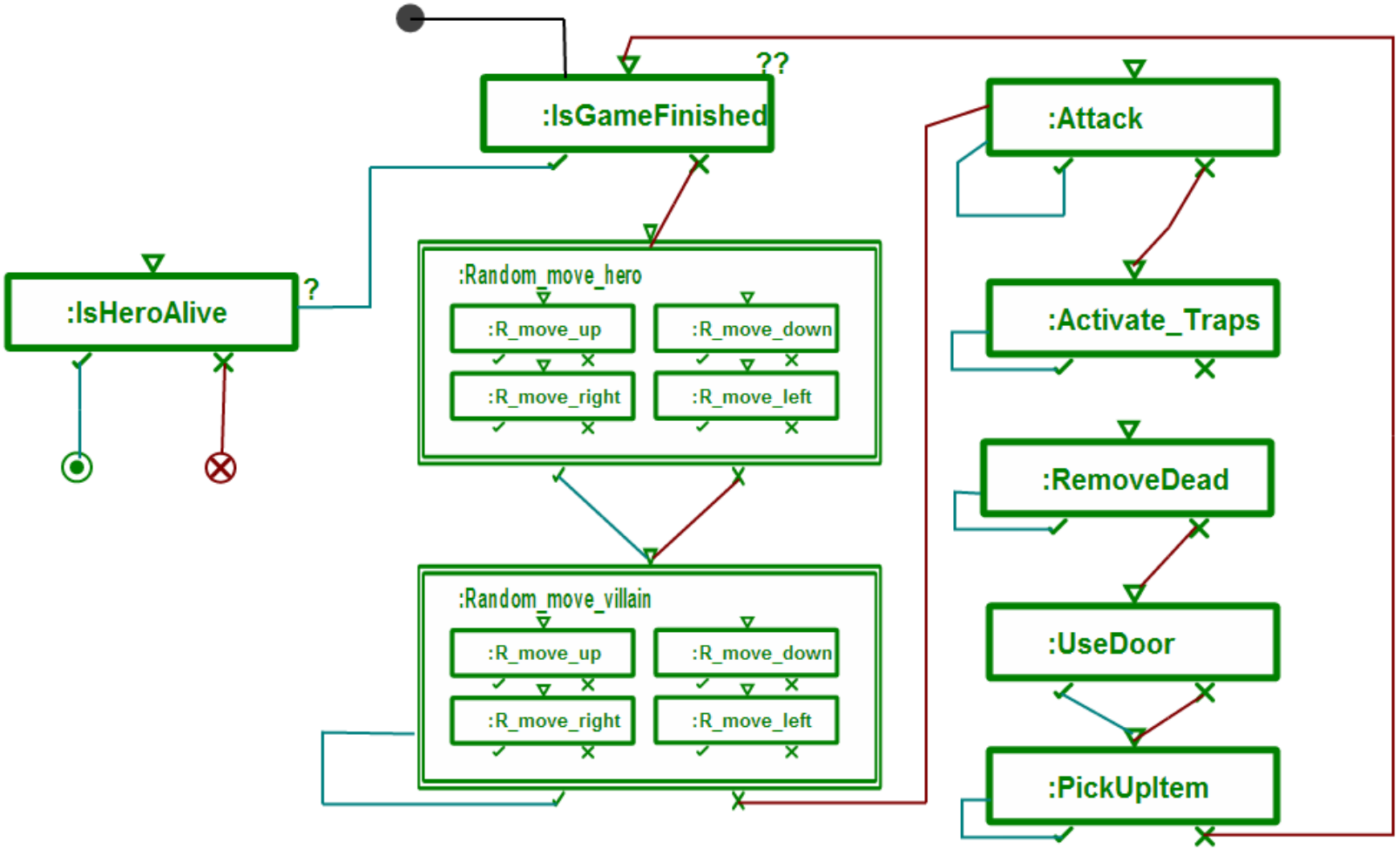
- Try DSL combination techniques hands-on
- Build test DSLs:
  - RPG (main)
  - Pathfinding (ext)
  - User Input (ext)
- Target language: Android

Table of Contents

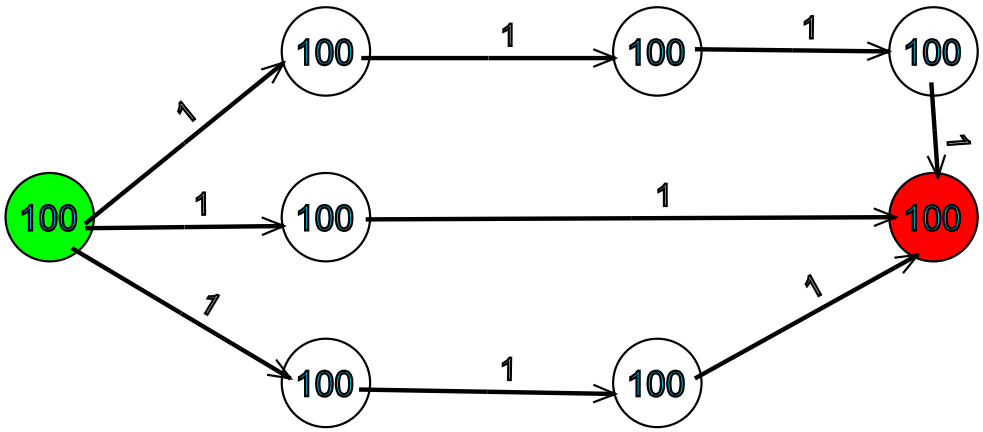
RPG



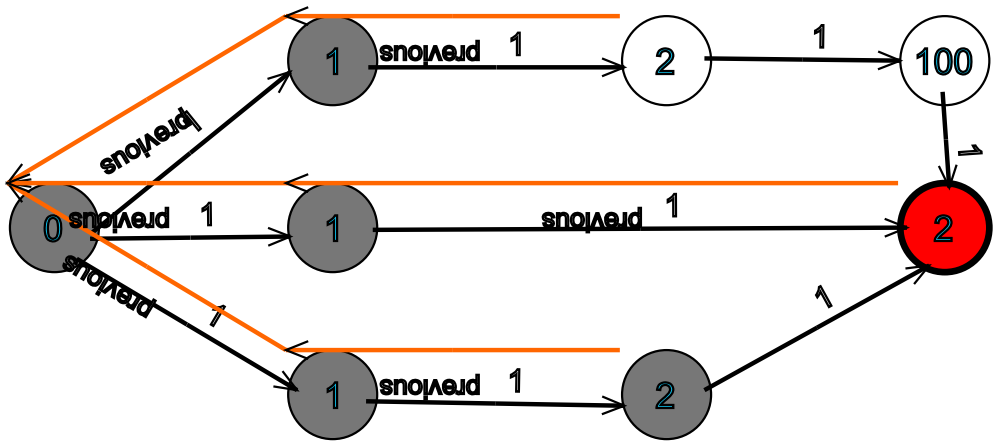
RPG


??
:IsGameFinished
:IsHeroAlive
?
:Random_move_hero
:R_move_up
:R_move_down
:R_move_right
:R_move_left
:Random_move_villain
:R_move_up
:R_move_down
:R_move_right
:R_move_left
:Attack
:Activate_Traps
:RemoveDead
:UseDoor
:PickUpItem

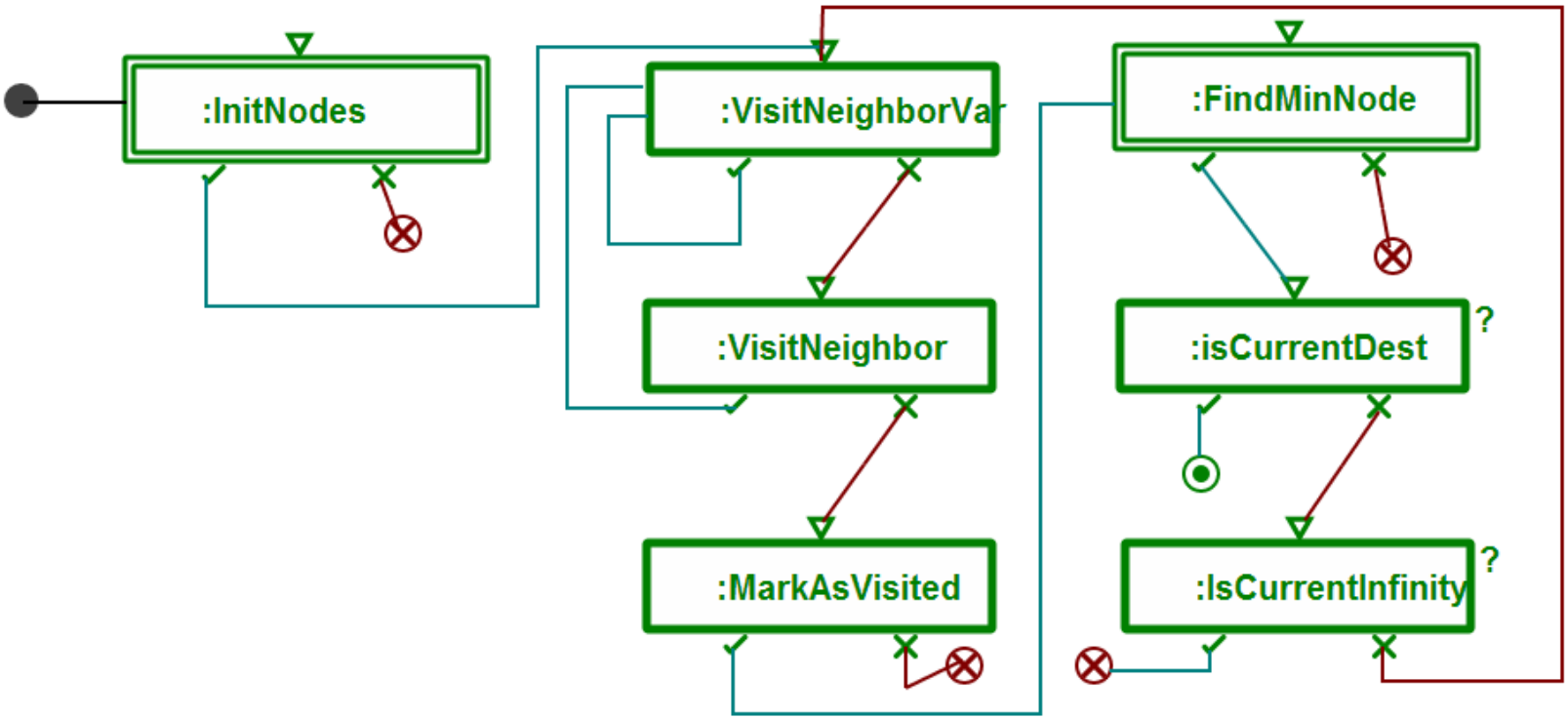
Pathfinding



Pathfinding

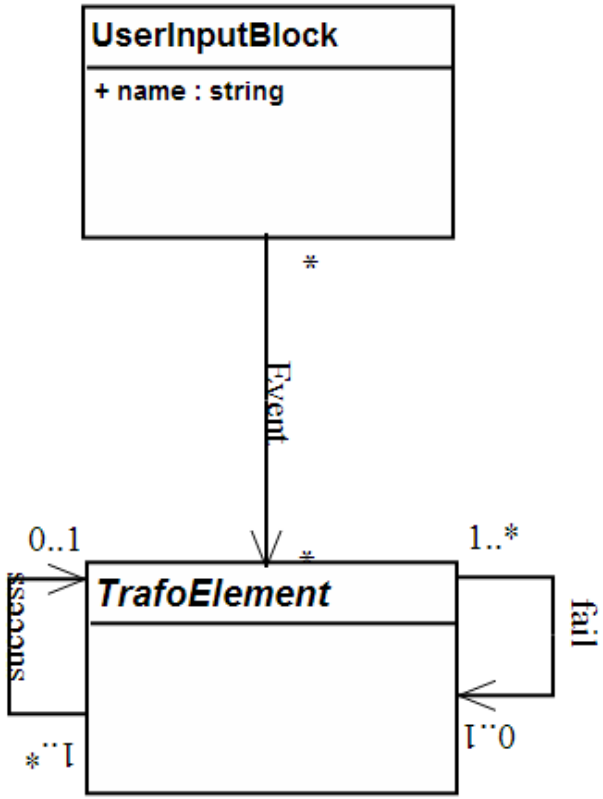


Pathfinding



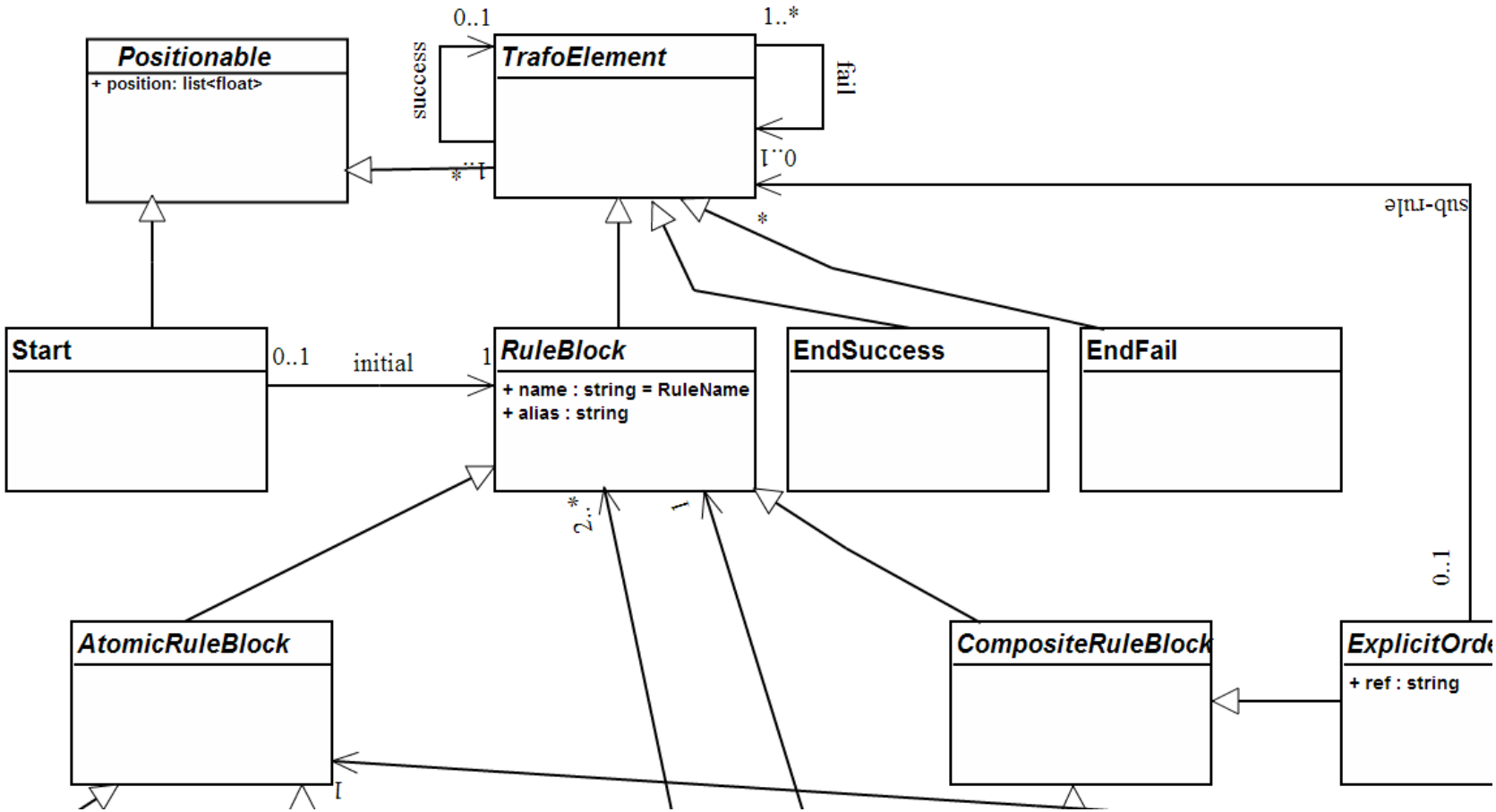
User Input

Abstract syntax



User Input

MoTif's abstract syntax



User Input

Instance

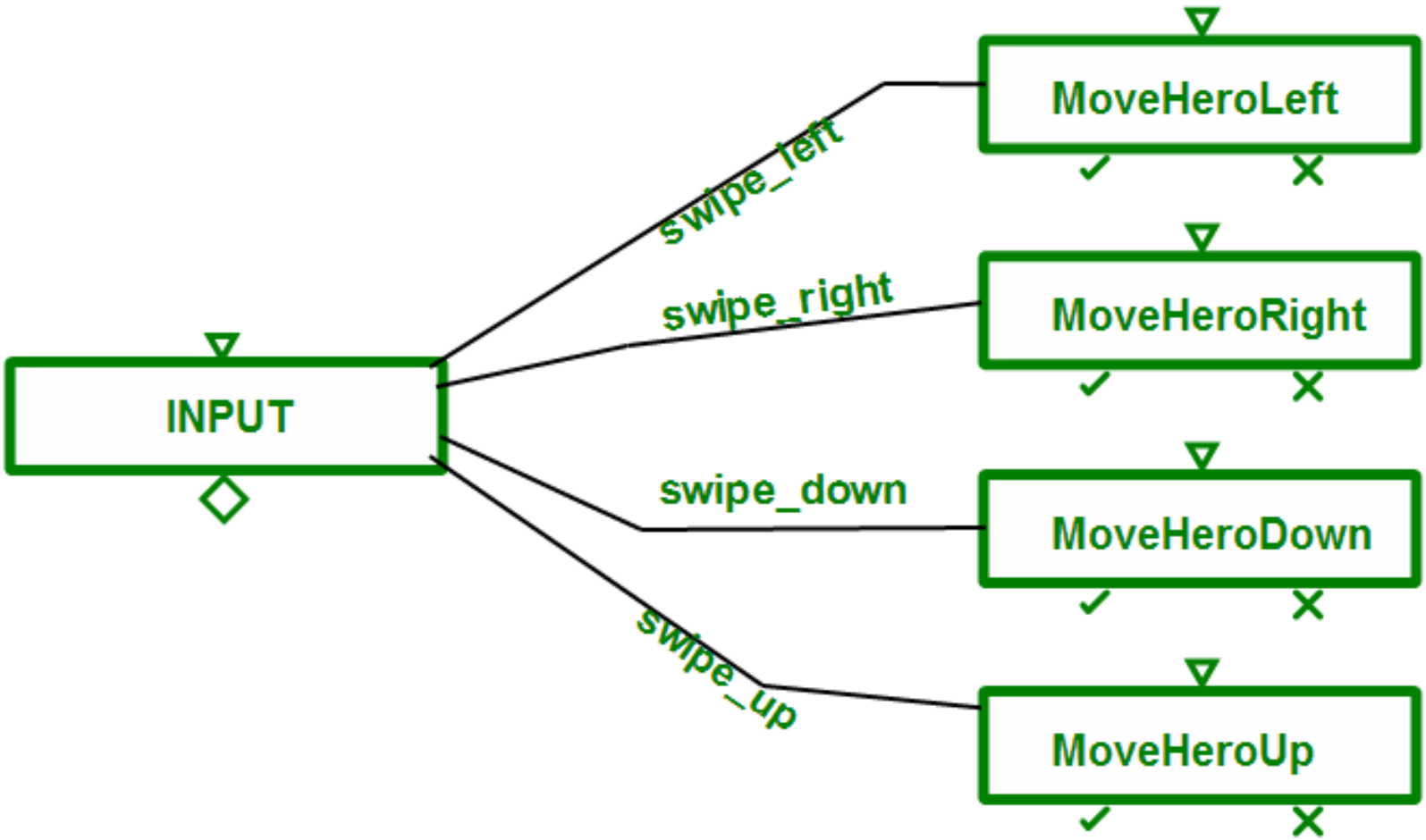
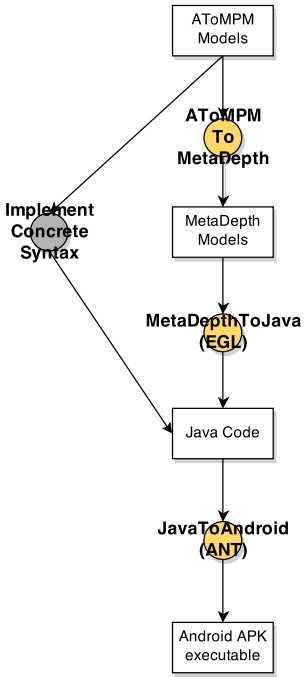
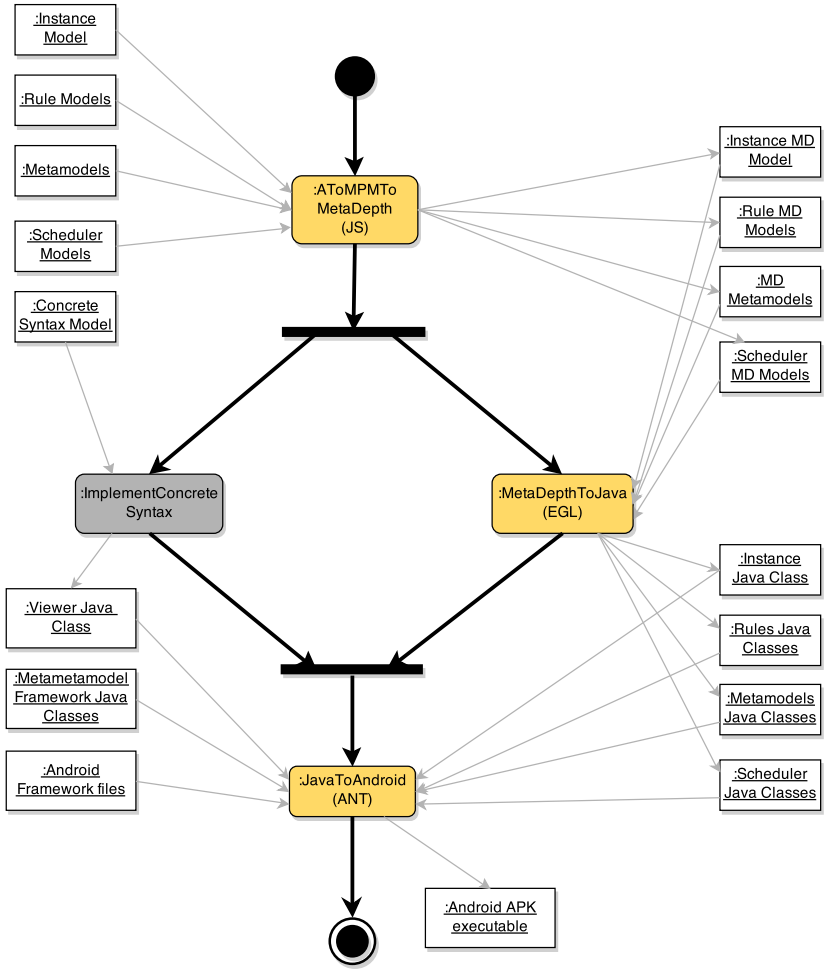


Table of Contents

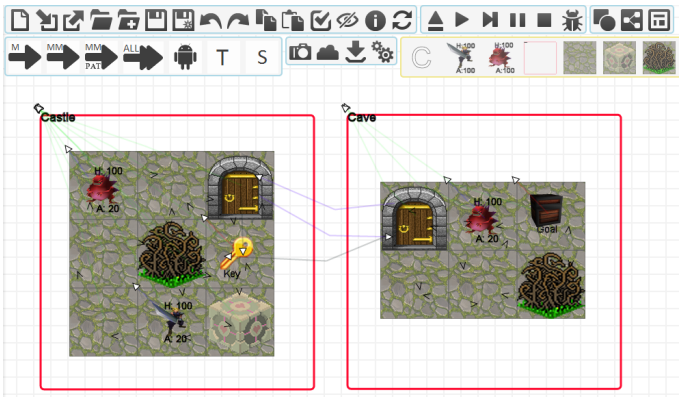
Code generation





AToMPM

- No links between links: association inheritance
- Concrete syntax not expressive enough
- Action/condition code limited to Python or JS (no modelled generic language)



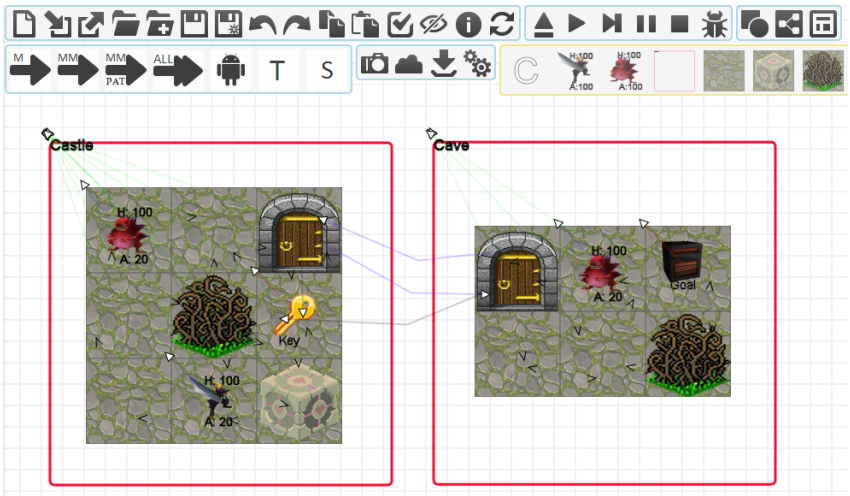
MetaDepth

- Support for EGL templates
- Extra transformation step: performance/bugs overhead
- Extend AToMPM to support EGL

```
> load "rpg"
:: loading collaborative/rpg.mdepth (25 clabjects created in 0.363 s).
> dump
:: dumping all
ext Model rpg@1{
  ext Node Link@1
  {
    dst@1: Node;
    src@1: Node;
  }
  ext Node Positionable@1
  {
    position@1:double[*]= [0.0, 0.0];
  }
  ext Node Item@1 : Positionable
  {
    name@1:String="item";
  }
  ext Node Key@1 : Item
  {
  }
  ext Node Weapon@1 : Item
  {
    attack@1:int=1;
  }
  ext Node Goal@1 : Item
  {
  }
  ext Node Character@1 : Positionable
  {
    hp@1:int=100;
    name@1:String;
    attack@1:int=20;
  }
  ext Node Villain@1 : Character
  {
  }
  ext Node Hero@1 : Character
  {
  }
  ext Node Tile@1 : Positionable
  {
  }
  ext Node Door@1 : Tile
  {
    isLocked@1:boolean=false;
  }
  ext Node Trap@1 : Tile
  {
    attack@1:int=1;
  }
  ext Node Obstacle@1 : Tile
  {
  }
  ext Node Scene@1 : Positionable
  {
    name@1:String;
```

AToMPM→MetaDepth: Javascript

- Extended export-to-metadepth plugin
- Automated export process
- All transformations called from AToMPM with one click



→

```
> load "rpg"
:: loading collaborative/rpg.mdepth (25 clabjects created in 0.363 s).
> dump
:: dumping all
ext Model rpg@1{
  ext Node Link@1
  {
    dst@1: Node;
    src@1: Node;
  }
  ext Node Positionable@1
  {
    position@1:double[*]= [0.0, 0.0];
  }
  ext Node Item@1 : Positionable
  {
    name@1:String="item";
  }
  ext Node Key@1 : Item
  {
  }
  ext Node Weapon@1 : Item
  {
    attack@1:int=1;
  }
  ext Node Goal@1 : Item
  {
  }
  ext Node Character@1 : Positionable
```

Java

No multiple inheritance (Possible in AToMPM and MetaDepth)

Simulate inheritance:

- Build subclass tree containing inheritance data
- No polymorphism
- Java Reflection for accessing attributes

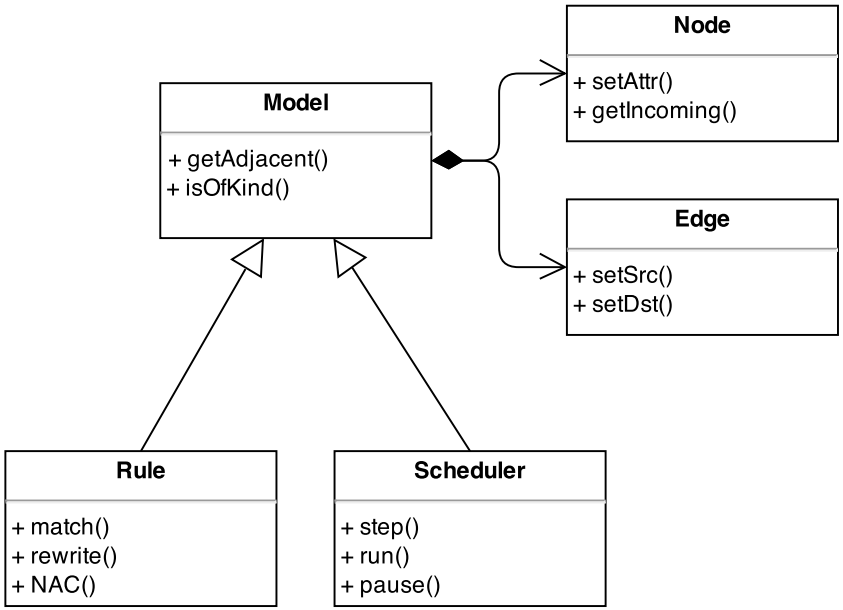
Rpg.java

```
package formalisms.rpg;

import formalisms.metametamodel.Model;

// The main model of the metamodel
public class Rpg extends Model {

    // Constructor
    public Rpg() {
        super();
        addSubclass("Link", "ConnectedToDoor");
        addSubclass("Link", "UnlocksDoor");
        addSubclass("Link", "CharToItem");
        addSubclass("Link", "TileToItem");
        addSubclass("Link", "Bottom");
        addSubclass("Link", "Top");
        addSubclass("Link", "Right");
        addSubclass("Link", "Left");
        addSubclass("Link", "TileToChar");
        addSubclass("Link", "SceneToTile");
        addSubclass("Positionable", "Item");
        addSubclass("Positionable", "Character");
        addSubclass("Positionable", "Tile");
        addSubclass("Positionable", "Scene");
        addSubclass("Item", "Key");
        addSubclass("Item", "Weapon");
        addSubclass("Item", "Goal");
        addSubclass("Key");
        addSubclass("Weapon");
        addSubclass("Goal");
        addSubclass("Character", "Villain");
        addSubclass("Character", "Hero");
        addSubclass("Villain");
        addSubclass("Hero");
        addSubclass("Tile", "Door");
        addSubclass("Tile", "Trap");
```

Java

Framework: Metametamodel

Query and manipulate generated models



MetaDepth→Java: EGL

- Metamodel
- Instance model
- Transformation rule
- Transformation scheduler

```
> load "rpg"
:: loading collaborative/rpg.mdepth (25 clabjects created in 0.363 s).
> dump
:: dumping all
ext Model rpg@1{
  ext Node Link@1
  {
    dst@1: Node;
    src@1: Node;
  }
  ext Node Positionable@1
  {
    position@1:double[*]= [0.0, 0.0];
  }
  ext Node Item@1 : Positionable
  {
    name@1:String="item";
  }
  ext Node Key@1 : Item
  {
  }
  ext Node Weapon@1 : Item
```

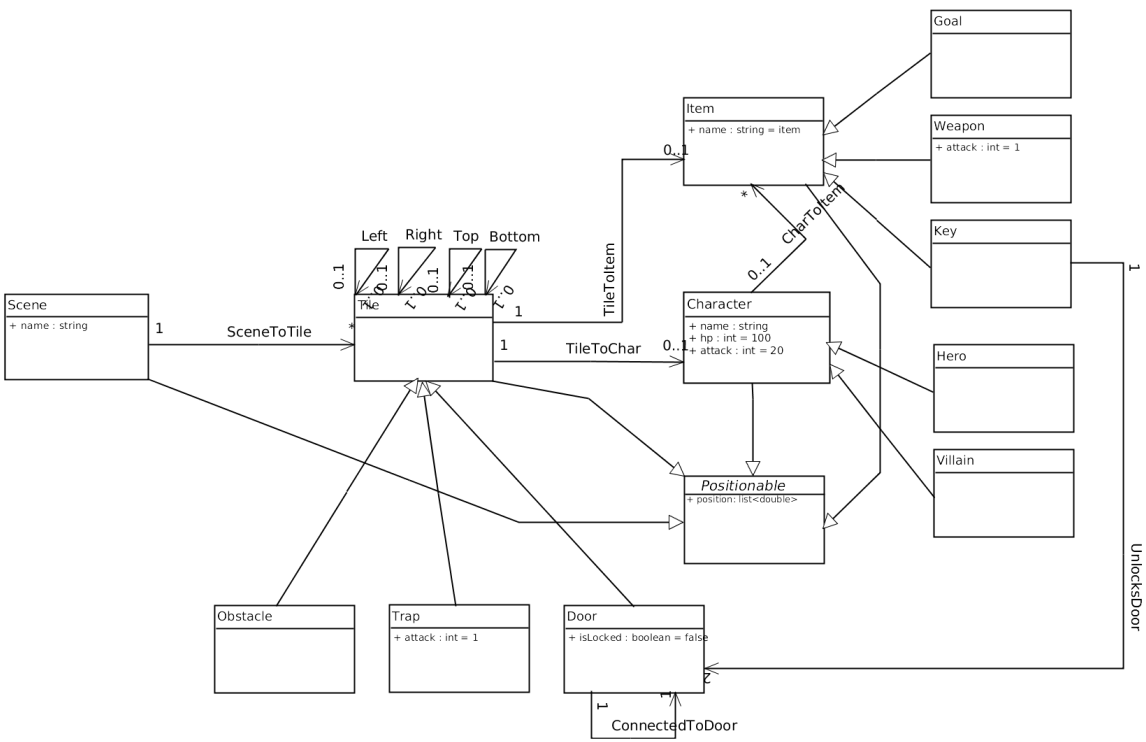
→

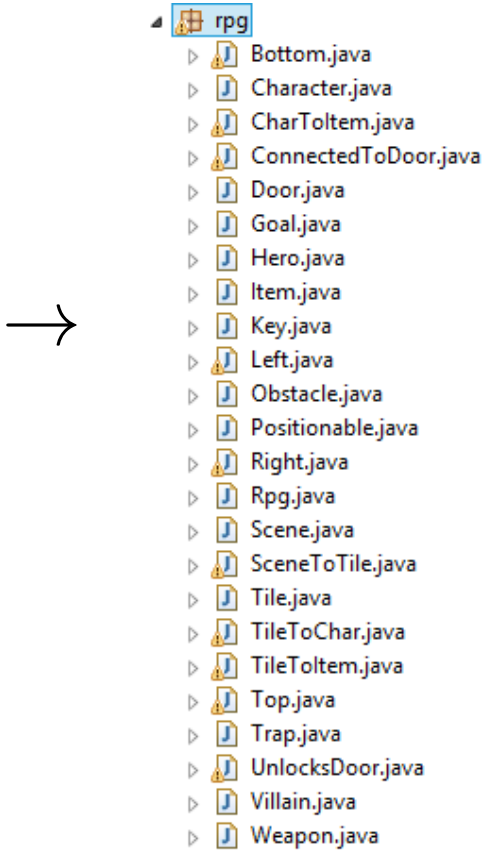
Rpg.java

```
package formalisms.rpg;

import formalisms.metametamodel.Model;

// The main model of the metamodel
public class Rpg extends Model {

    // Constructor
    public Rpg() {
        super();
        addSubclass("Link", "ConnectedToDoor");
        addSubclass("Link", "UnlocksDoor");
        addSubclass("Link", "CharToItem");
```

Metamodel

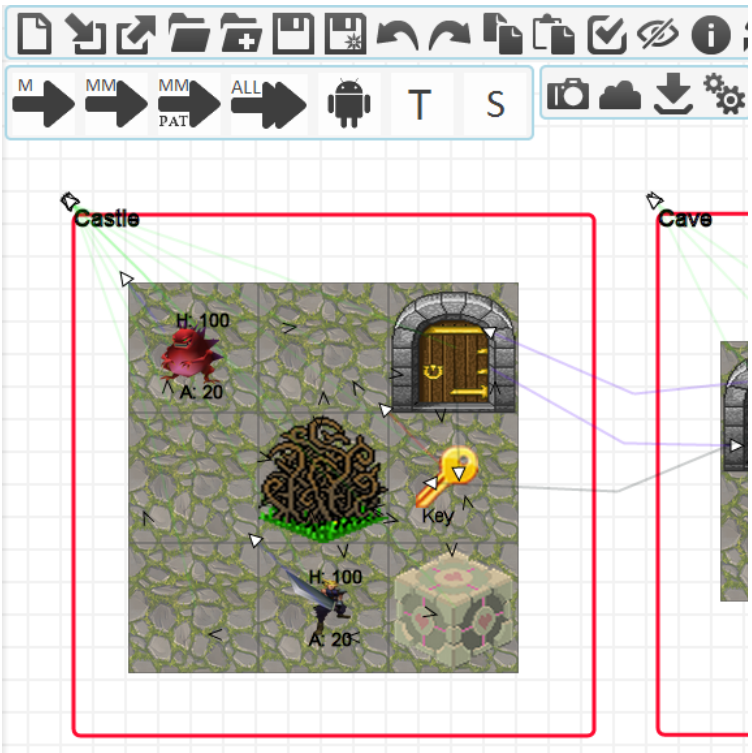
Generated metamodel code

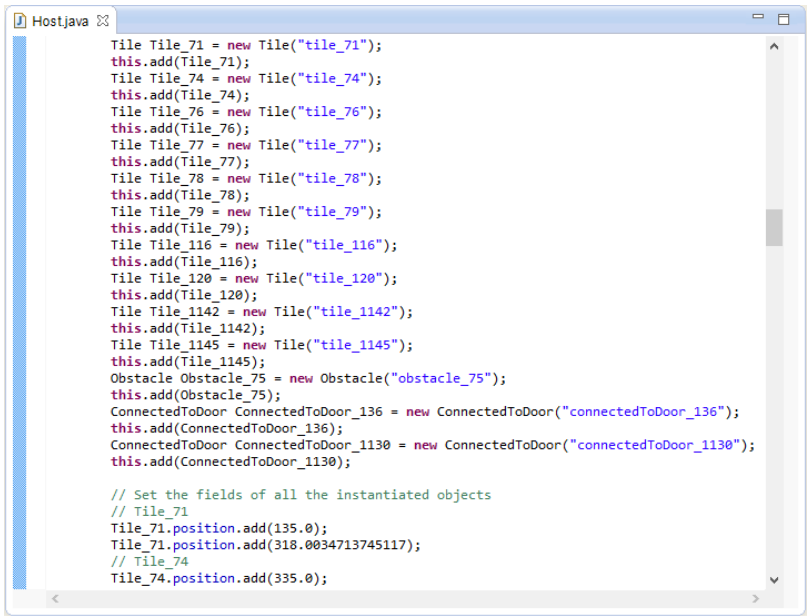




Instance model

Generated instance model code



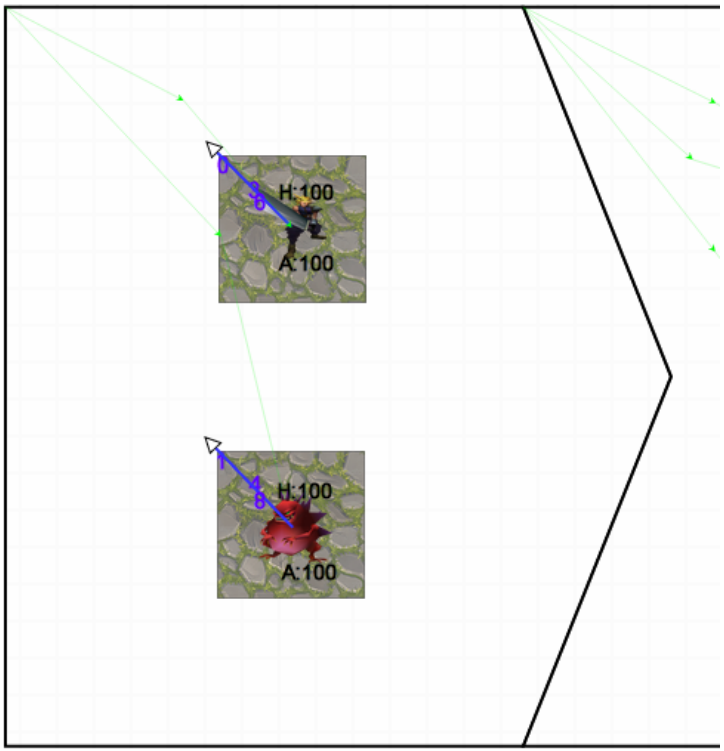


Host.java

```
Tile Tile_71 = new Tile("tile_71");
this.add(Tile_71);
Tile Tile_74 = new Tile("tile_74");
this.add(Tile_74);
Tile Tile_76 = new Tile("tile_76");
this.add(Tile_76);
Tile Tile_77 = new Tile("tile_77");
this.add(Tile_77);
Tile Tile_78 = new Tile("tile_78");
this.add(Tile_78);
Tile Tile_79 = new Tile("tile_79");
this.add(Tile_79);
Tile Tile_116 = new Tile("tile_116");
this.add(Tile_116);
Tile Tile_120 = new Tile("tile_120");
this.add(Tile_120);
Tile Tile_1142 = new Tile("tile_1142");
this.add(Tile_1142);
Tile Tile_1145 = new Tile("tile_1145");
this.add(Tile_1145);
Obstacle Obstacle_75 = new Obstacle("obstacle_75");
this.add(Obstacle_75);
ConnectedToDoor ConnectedToDoor_136 = new ConnectedToDoor("connectedToDoor_136");
this.add(ConnectedToDoor_136);
ConnectedToDoor ConnectedToDoor_1130 = new ConnectedToDoor("connectedToDoor_1130");
this.add(ConnectedToDoor_1130);

// Set the fields of all the instantiated objects
// Tile_71
Tile_71.position.add(135.0);
Tile_71.position.add(318.0034713745117);
// Tile_74
Tile_74.position.add(335.0);
```

Rule matcher

Generated rule model code



R_kill_villain.java

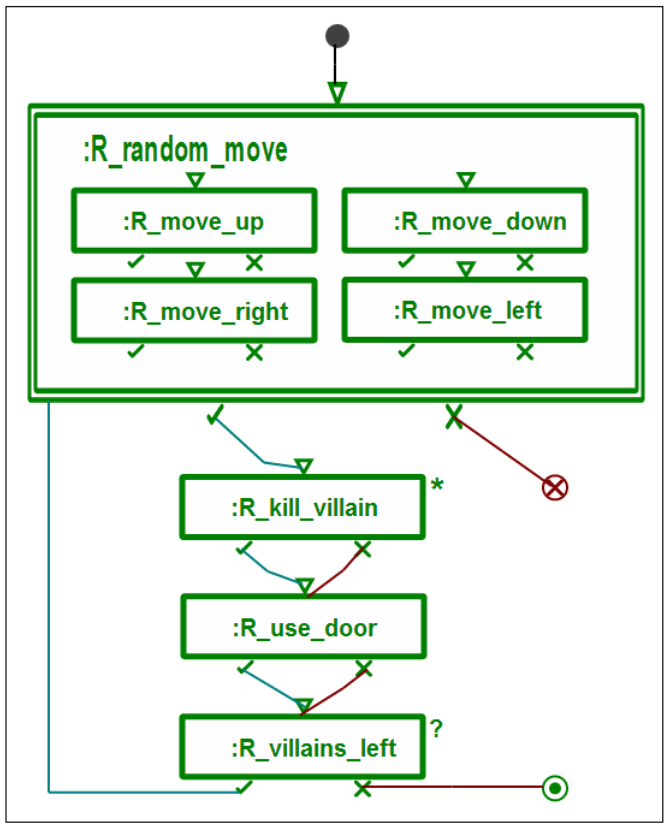
```
public class R_kill_villain extends RuleModel {

    // LHS_68
    public HashMap<Integer, Node> match(Model host) {
        HashMap<Integer, Node> match = new HashMap<Integer, Node>();
        ArrayList<Node> candidates;
        candidates = new ArrayList<Node>();
        candidates.addAll(host.getNodesOfKind("Tile"));
        for (Node __pTile_127 : candidates) {
            if (!match.containsValue(__pTile_127) && isOfKind("Tile", __pTile_127) && _
                match.put(0, __pTile_127);
                candidates = new ArrayList<Node>();
                candidates.addAll(host.getOutgoing(__pTile_127));
                for (Node __pTileToChar_133 : candidates) {
                    if (!match.containsValue(__pTileToChar_133) && isOfType("TileToChar
                        match.put(6, __pTileToChar_133);
                        candidates = new ArrayList<Node>();
                        candidates.addAll(host.getOutgoing(__pTileToChar_133));
                        for (Node __pHero_132 : candidates) {
                            if (!match.containsValue(__pHero_132) && isOfType("Hero", _
                                match.put(3, __pHero_132);
                                candidates = new ArrayList<Node>();
                                candidates.addAll(host.getNodesOfKind("Tile"));
                                for (Node __pTile_128 : candidates) {
                                    if (!match.containsValue(__pTile_128) && isOfKind("
                                        match.put(1, __pTile_128);
                                        candidates = new ArrayList<Node>();
                                        candidates.addAll(host.getOutgoing(__pTile_128)
                                        for (Node __pTileToChar_154 : candidates) {
                                            if (!match.containsValue(__pTileToChar_154)
                                                match.put(8, __pTileToChar_154);
                                                candidates = new ArrayList<Node>();
                                                candidates.addAll(host.getOutgoing(__pT
                                                for (Node __pVillain_153 : candidates)
                                                    if (!match.containsValue(__pVillain
                                                        match.put(4, __pVillain_153);
                                                        // Match successful, check glob
                                                        if(LHS_68_condition(match))
```

Rule matcher

Action/condition code

Rule Scheduler

Generated transformation scheduler code

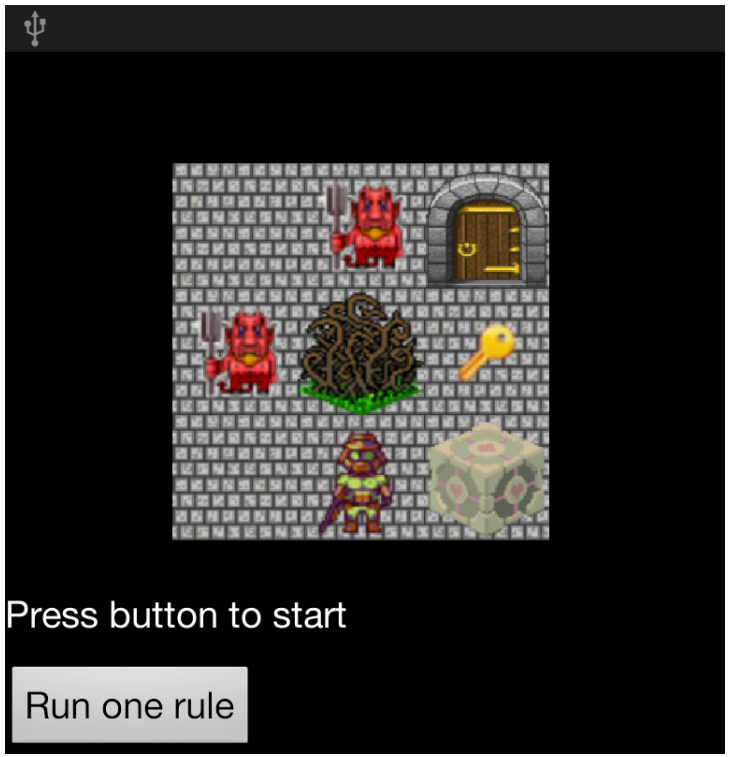


Schedule.java

```
@Override
public String step() {
    boolean success = false;
    switch (currentState) {
    case R_kill_villain:
        boolean currentSuccess_SRule_311 = false;
        for (int i = 0; i < 1000; i++) {
            currentSuccess_SRule_311 =
                    rules.get(State.R_kill_villain).applyRule(host) != null;
            success = success || currentSuccess_SRule_311;
            if (!currentSuccess_SRule_311)
                break;
        }
        if (success) {
            currentState = State.R_use_door;
            return "Rule succeeded (R_kill_villain)";
        } else {
            currentState = State.R_use_door;
            return "Rule failed (R_kill_villain)";
        }
    case R_random_move:
        Collections.shuffle(R_random_move_contents);
        for (int i = 0; i < R_random_move_contents.size(); i++) {
            RuleModel rule = R_random_move_contents.get(i);
            success = rule.applyRule(host) != null;
            if (success)
                break;
        }
        if (success) {
            currentState = State.R_kill_villain;
            return "Rule succeeded (R_random_move)";
        } else {
            return "Rule failed (R_random_move)";
        }
    case R_use_door:
        success = rules.get(State.R_use_door).applyRule(host) != null;
        if (success) {
```

Android
Framework

- Activity.java (app's entry point)
- layout.xml (user interface)
- View.java (model concrete syntax)



Java→Android: Ant

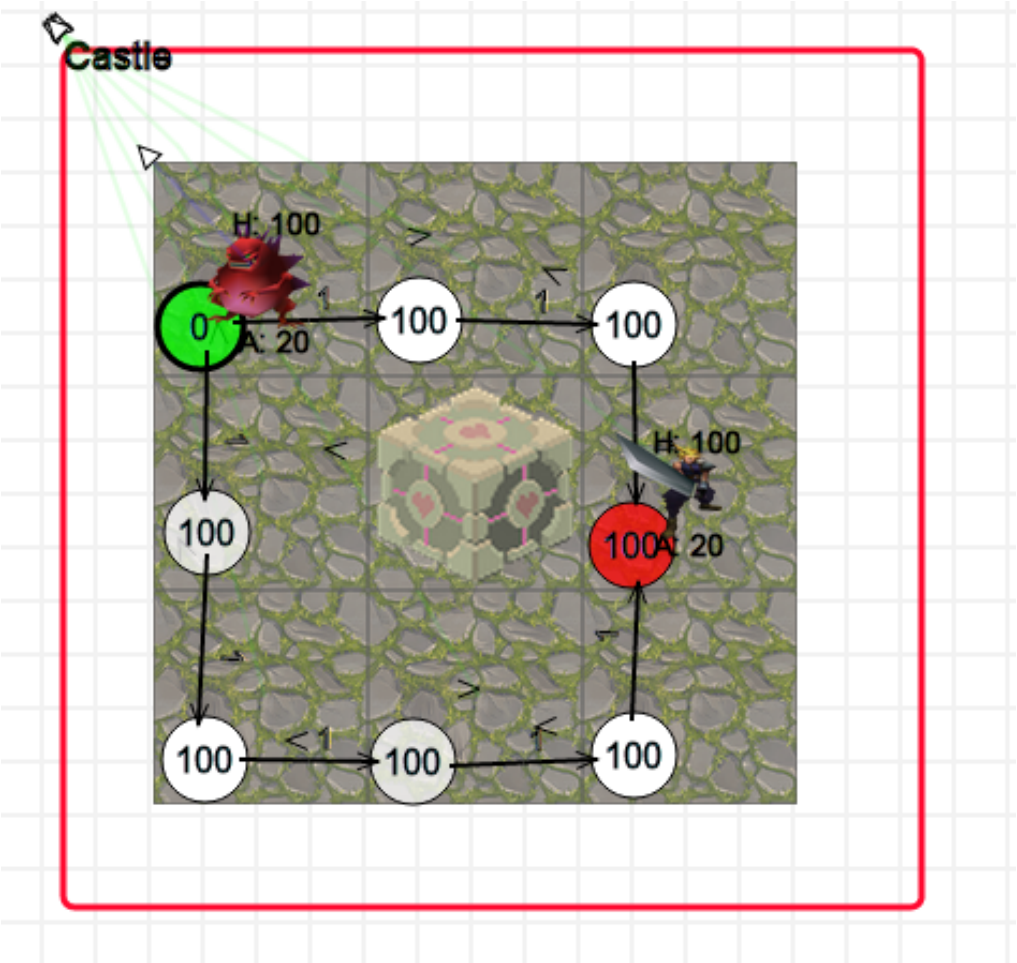
Tasks

- Compile Java files
- Build APK
- Install on device
- Run on device

```
    [java]   Sat Jun 28 15:47:46 CEST 2014
    [java] > :: setting environment variable DIR
    [java] > :: loading AToMPM2Android/metaDepth/rpg.mdepth (2!
    [java] > :: entering context rpg
    [java] > :: setting environment variable DIR
    [java] > Generated file: Positionable.java
    [java] Generated file: Item.java
    [java] Generated file: Key.java
    [java] Generated file: Weapon.java
    [java] Generated file: Goal.java
    [java] Generated file: Character.java
    [java] Generated file: Villain.java
    [java] Generated file: Hero.java
    [java] Generated file: Tile.java
    [java] Generated file: Door.java
    [java] Generated file: Trap.java
    [java] Generated file: Obstacle.java
    [java] Generated file: Scene.java
    [java] Generated file: ConnectedToDoor.java
    [java] Generated file: UnlocksDoor.java
    [java] Generated file: CharToItem.java
    [java] Generated file: TileToItem.java
    [java] Generated file: Bottom.java
    [java] Generated file: Top.java
    [java] Generated file: Right.java
    [java] Generated file: Left.java
    [java] Generated file: TileToChar.java
    [java] Generated file: SceneToTile.java
    [java] Generated file: Rpg.java
    [java] :: loading AToMPM2Android/EGL/src/mm/mm_main.egl
    [java] > :: bye!
    [java] MetaDepth to Java Generation completed successfully
   [javac] Compiling 9 source files to D:\Dropbox\UA\RI2\AToMP!
BUILD SUCCESSFUL
Total time: 21 seconds
```

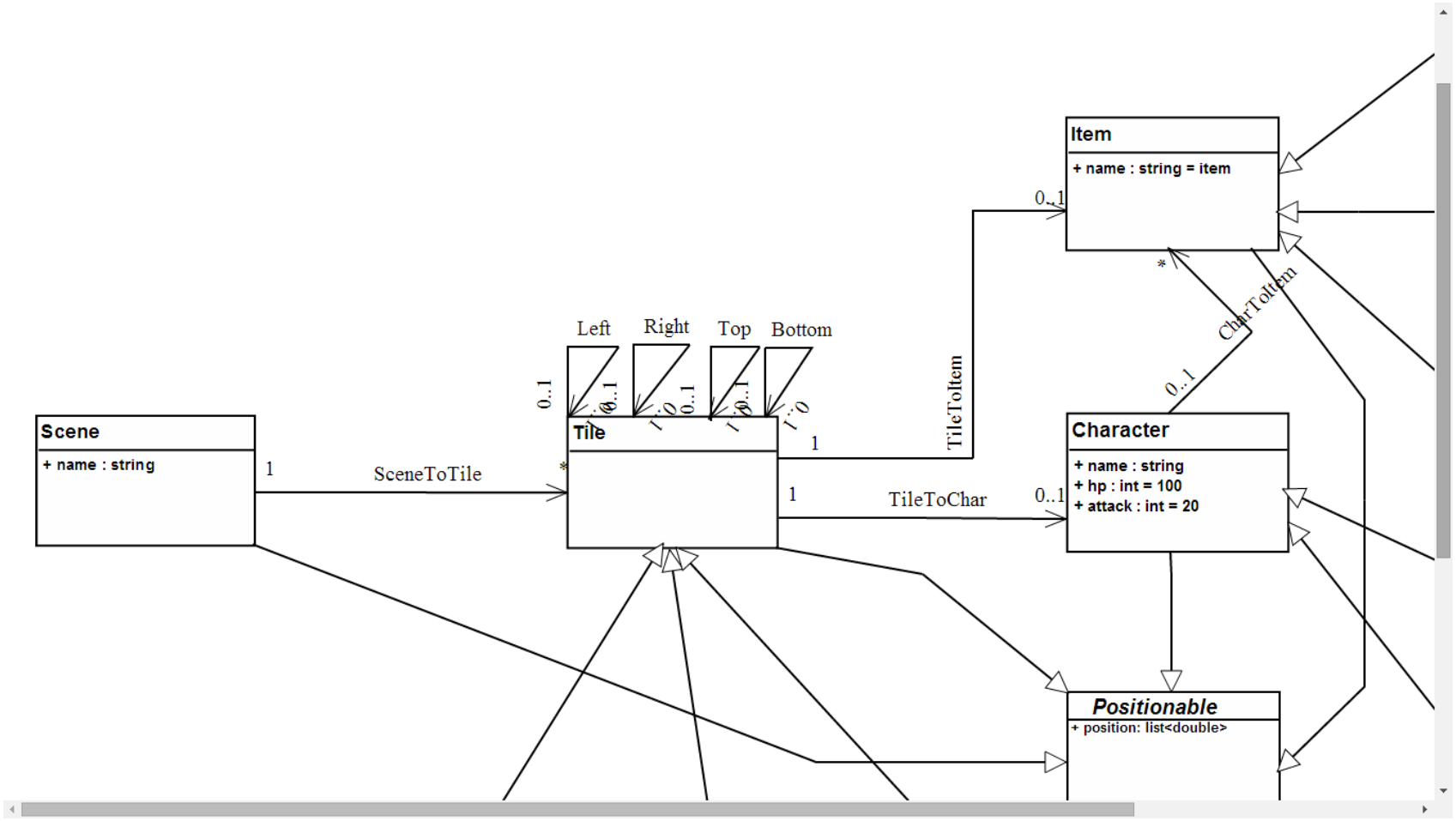
Table of Contents

RPG + Pathfinding



RPG MM

Before weaving



RPG MM

After weaving by inheritance