Stavros Vassos, University of Athens, Greece stavrosv@di.uoa.gr May 2012

INTRODUCTION TO AI STRIPS PLANNING

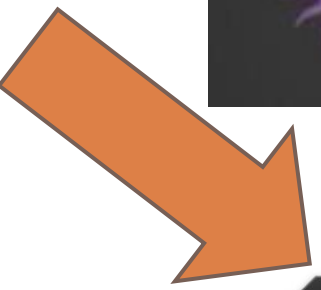
.. and Applications to Video-games!

Course overview

- Lecture 1: Game-inspired competitions for AI research, AI decision making for non-player characters in games
- Lecture 2: STRIPS planning, state-space search
- Lecture 3: Planning Domain Definition Language (PDDL), using an award winning planner to solve Sokoban
- Lecture 4: Planning graphs, domain independent heuristics for STRIPS planning
- Lecture 5: Employing STRIPS planning in games: SimpleFPS, **iThinkUnity3D**, **SmartWorkersRTS**
- Lecture 6: Planning beyond STRIPS

STRIPS in a real game engine

- Amazing tools available for (indie) game developers!



ShiVa3D

UDK

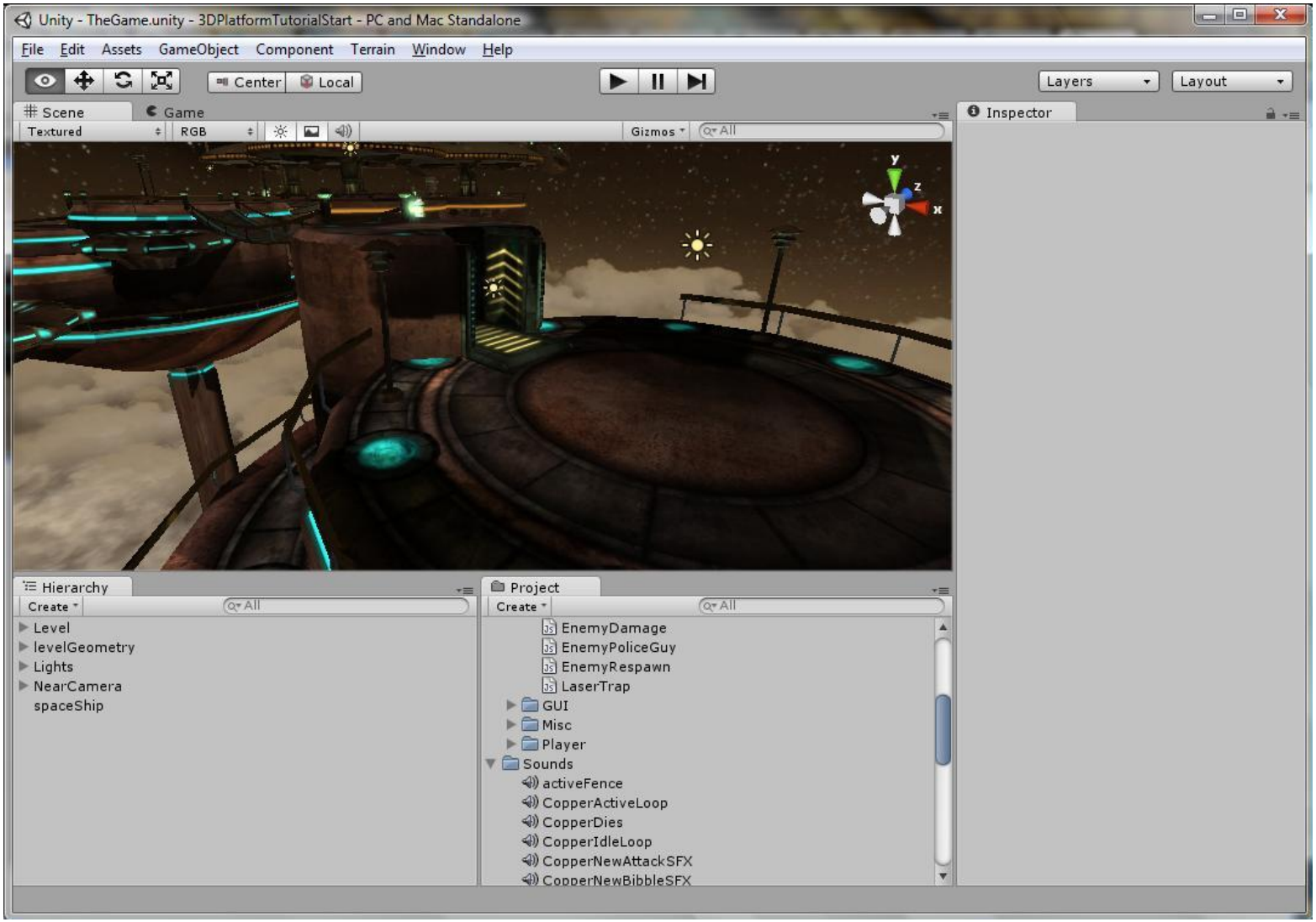
unity

VALVE

Game development with Unity3D

- Integrated Game Development Environment
- C#, Javascript, Boo programming languages
- Asset-centric instead of code-centric, adopting a look and feel like 3D CAD software

Game development with Unity3D



Game development with Unity3D

- Terminology
  - Project
  - Scene
  - GameObject and Component
  - Asset and Prefab
  - Script

Game development with Unity3D

- 3D platform game tutorial available online by Unity3D
  - http://unity3d.com/support/resources/tutorials/3d-platform-game

Game development with Unity3D

- Sections 1,2 of the tutorial
  - Start with an empty platform level
  - Add our player: Lerpz
  - Add a camera that follows him
  - Add a 3rd person controller to control Lerpz
  - Tweak his movement
- Section 5
  - Add NPCs!

Game development with Unity3D

- Quick demo using
  - Lerpz
  - SpringFollowCamera
  - ThirdPersonController
  - CharacterController
  - ThirdPersonPlayerAnimation

Game development with Unity3D

Game development with Unity3D

- Unity3D
  - Provides a basic simulated environment to build AI agents
  - Can be used as an educational platform to experiment with AI techniques about knowledge representation, reasoning, agent languages and systems, robotics, …
  - Can be used as a realistic test-bed to try AI techniques for NPCs in commercial video-games

iThink: STRIPS planning in Unity3D

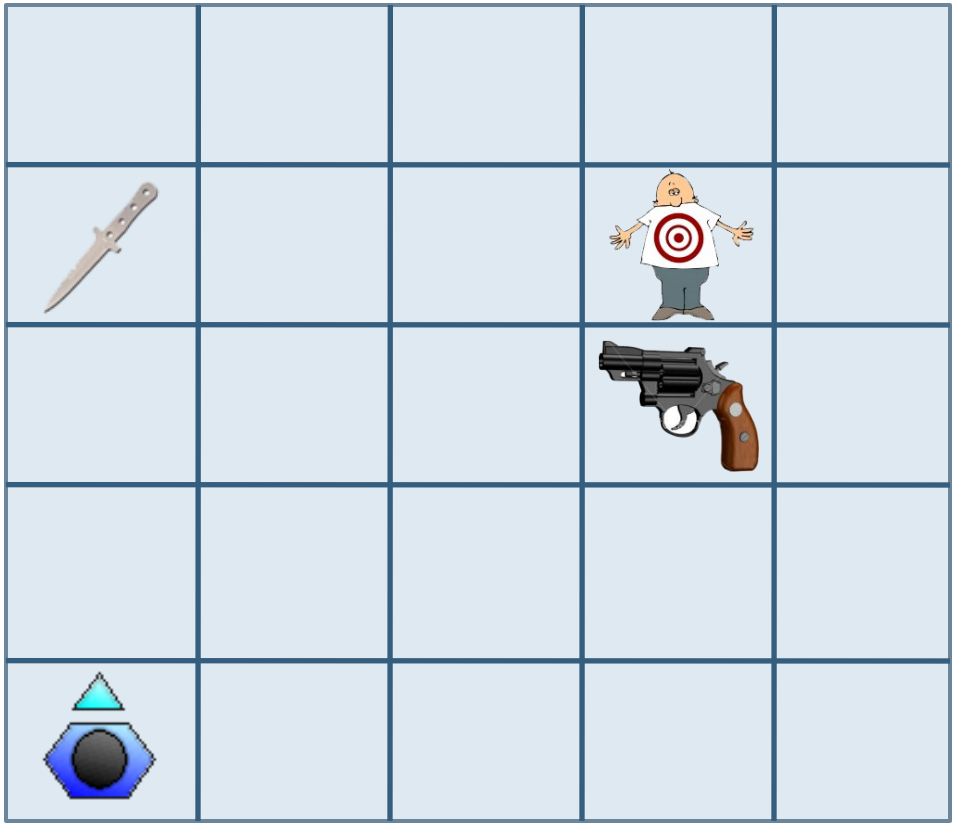
- B.Sc. project at the University of Athens
  - Vassileios-Marios Anastassiou
  - Panagiotis Diamantopoulos
- SETN-2012 conference paper
  - iThink: A Library for Classical Planning in Video-games
- Code available online
  - https://code.google.com/p/ithink-unity3d/

iThink: STRIPS planning in Unity3D

- iThink-Unity3D
  - Provides a basic framework for specifying and solving STRIPS planning problems inside Unity3D
  - Can be used as an educational platform to experiment with STRIPS planning and extensions
  - Can be used as a realistic test-bed to try STRIPS planning in commercial games

iThink: STRIPS planning in Unity3D

- SimpleGame domain



- turn(?fromd ?tod)
- move(?froml ?tol ?dir)
- pickup(?o ?l)
- stab(?l ?knife)
- shoot(?locn ?locp ?dir ?gun)

iThink: STRIPS planning in Unity3D

```
// Defining a STRIPS action schema

class ActionSGMove : iThinkAction
{
    GameObject From, To, Dir;
    public ActionSGMove(      string name,
                              GameObject from,
                              GameObject to,
                              GameObject dir )
        : base( name )
    {
        From = from;
        To = to;
        Dir = dir;

        initPreconditions();
        initEffects();
    }
```

iThink: STRIPS planning in Unity3D

```
    public override void initPreconditions()
    {
        base.initPreconditions();
        preconditions.Add( new iThinkFact( "npcAt", From ) );
        preconditions.Add( new iThinkFact( "npcFacing", Dir ) );
        preconditions.Add( new iThinkFact( "adjacent",
                                           From,
                                           To,
                                           Dir ) );
    }

    public override void initEffects()
    {
        base.initEffects();
        effects.Add( new iThinkFact( "npcAt", To ) );
        effects.Add( new iThinkFact( "npcAt", false, From ) );
    }
}
```

iThink: STRIPS planning in Unity3D

```
// Defining the initial state

factList = new List<iThinkFact>();

factList.Add( new iThinkFact( "npcAt",
                              GameObject.Find( "LOC1" ) ) );

factList.Add( new iThinkFact( "npcFacing",
                              GameObject.Find( "UP" ) ) );

factList.Add( new iThinkFact( "npcEmptyHands" ) );

factList.Add( new iThinkFact( "playerAt",
                              GameObject.Find( "LOC8" ) ) );
...

 brain.startState = new iThinkState( "Initial",
             new List<iThinkFact>( factList ) );
```

iThink: STRIPS planning in Unity3D

```
// Defining the goal state

goalfactList = new List<iThinkFact>();

goalfactList.Add( new iThinkFact( "playerDown" ) );


brain.goalState = new iThinkState( "Goal",
            new List<iThinkFact>( goalfactList ) );
```

iThink: STRIPS planning in Unity3D

```
// Start planning!


 // Specify search method
brain.planner.forwardSearch (brain.startState ,
                             brain.goalState ,
                             brain.ActionManager ,
                             1);


// Get the plan as a sequence of actions
brain.planner.getPlan().debugPrintPlan();
```

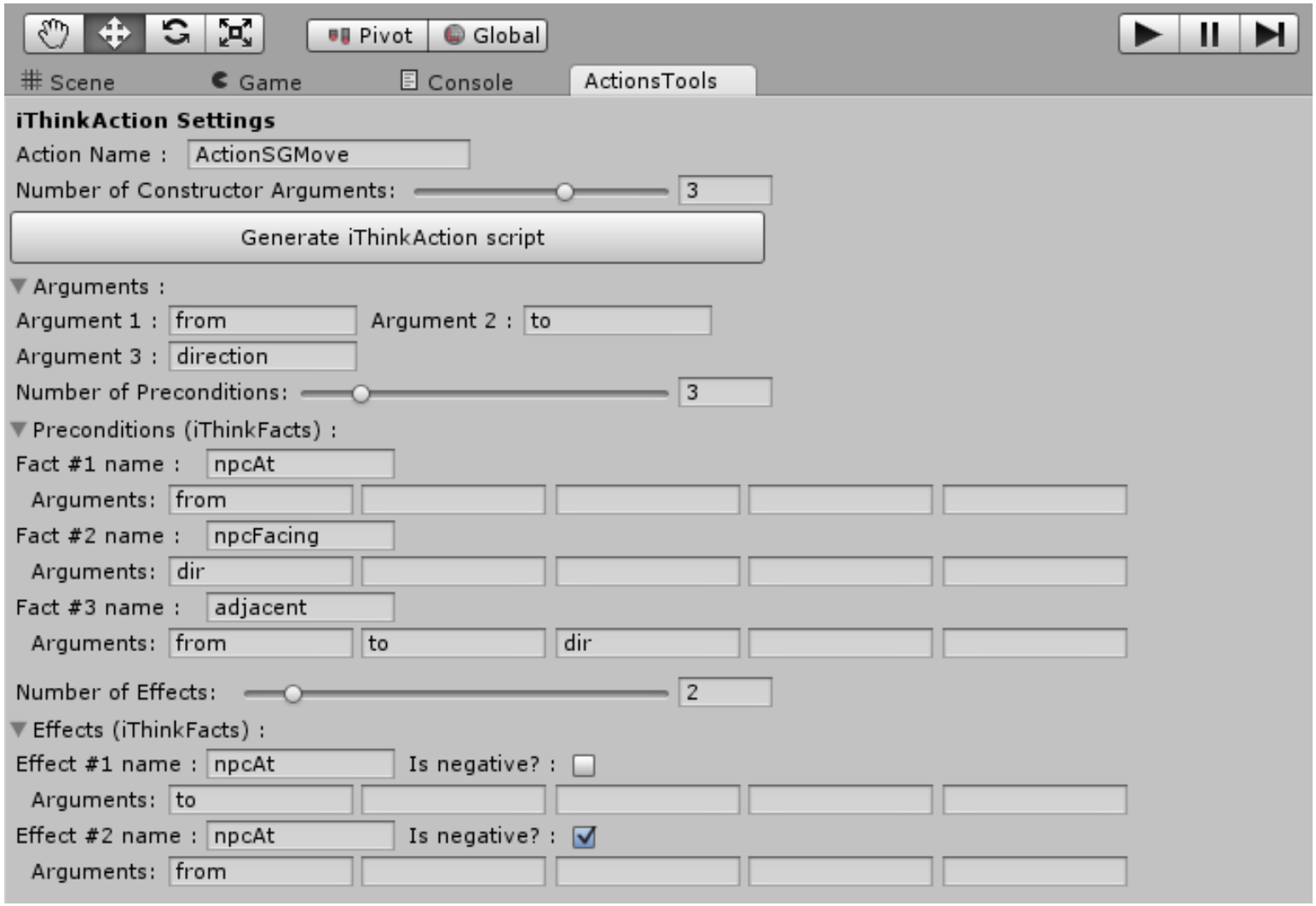
iThink: STRIPS planning in Unity3D

- iThink-Unity3D
- iThinkBrain uses several classes and components
  - Fact, State, Action, Plan, Planner
  - SensorySystem
  - ActionManager, ActionSchemas
- Basic search methods implemented
  - Depth-First, Breadth-First, Best-First, A*
  - A modular design allows to easily integrate different methods

iThink: STRIPS planning in Unity3D

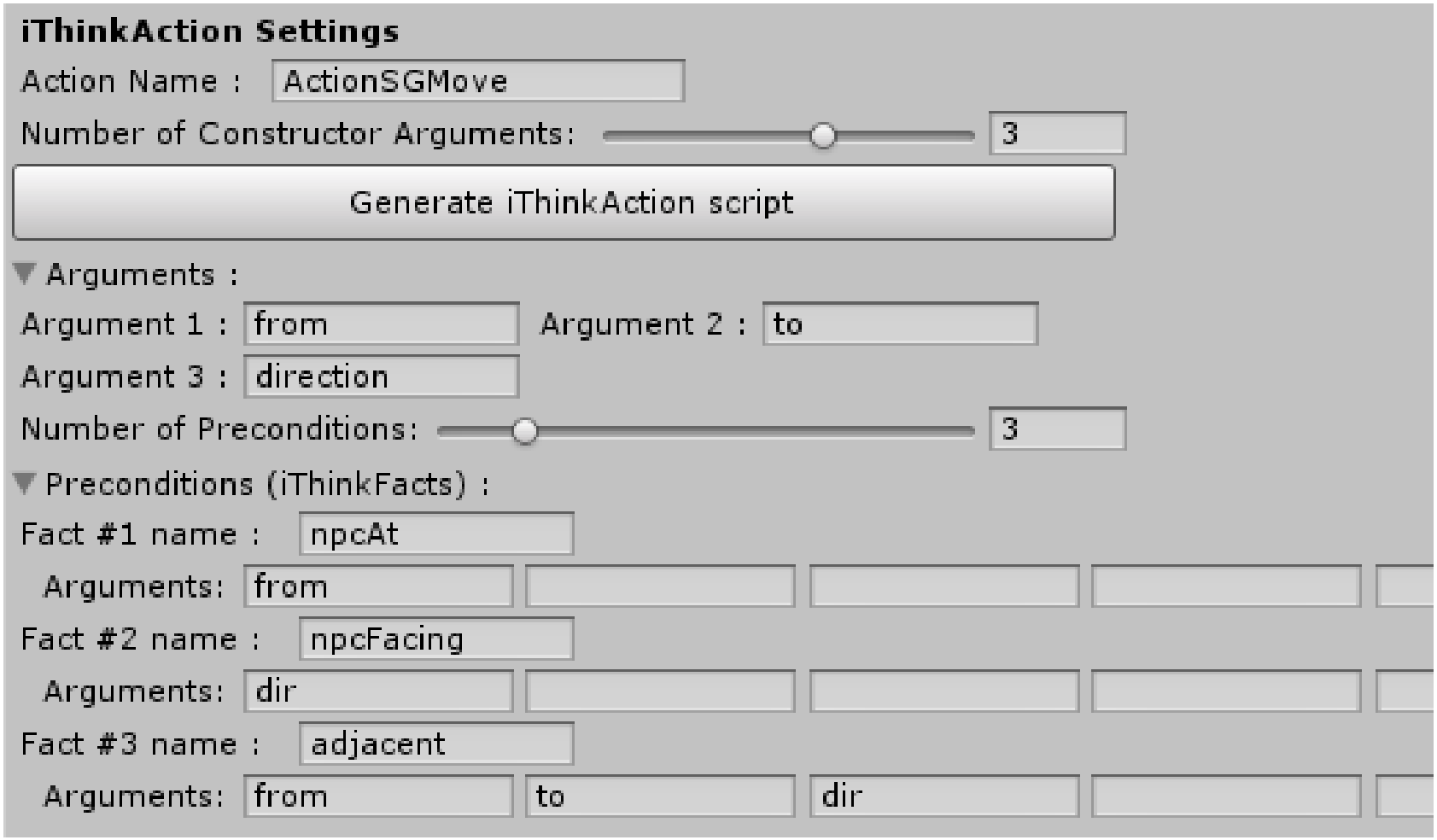
- iThink-Unity3D
- GUI under development!

iThink: STRIPS planning in Unity3D



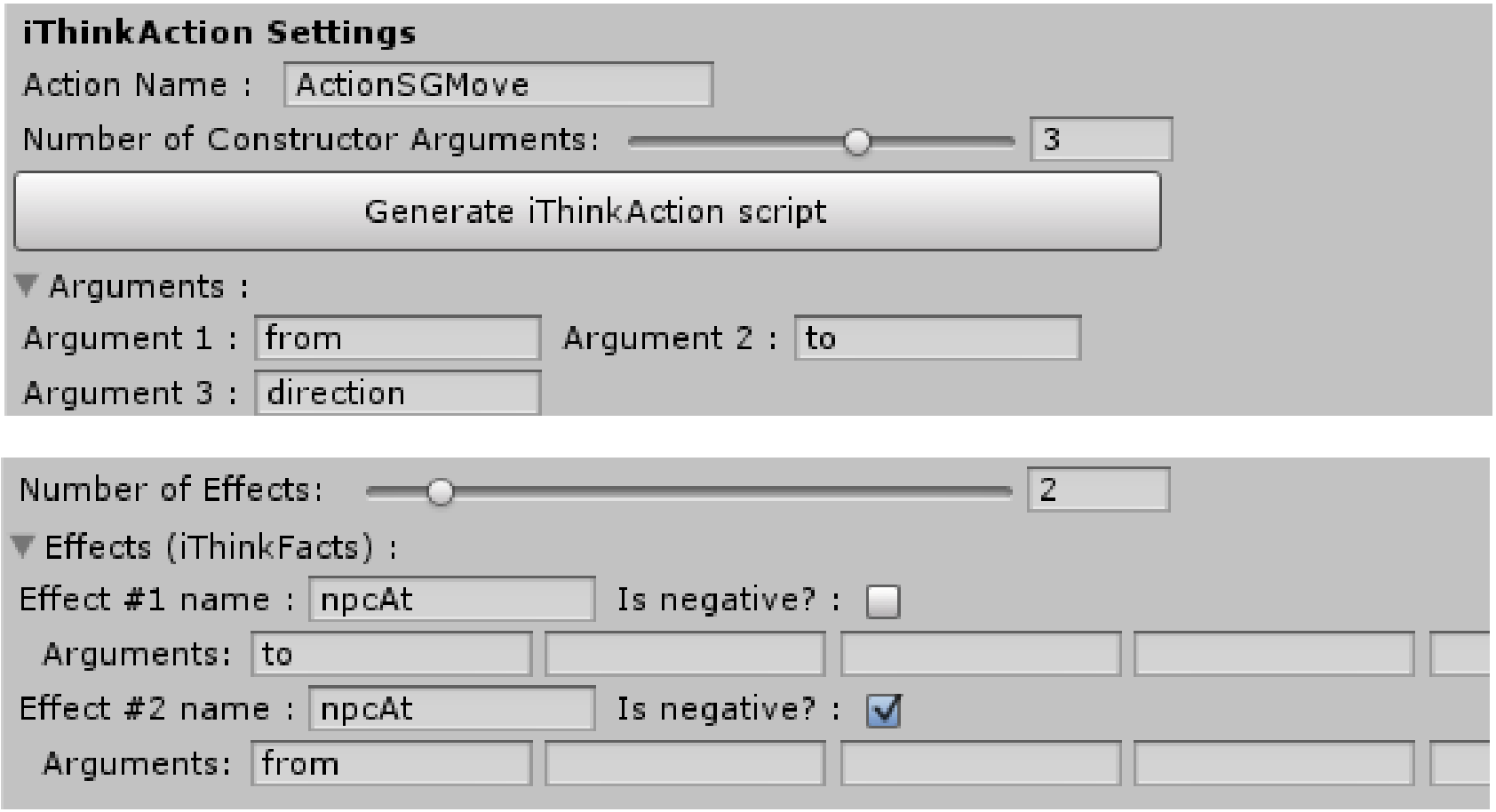
iThink: STRIPS planning in Unity3D

- Preconditions



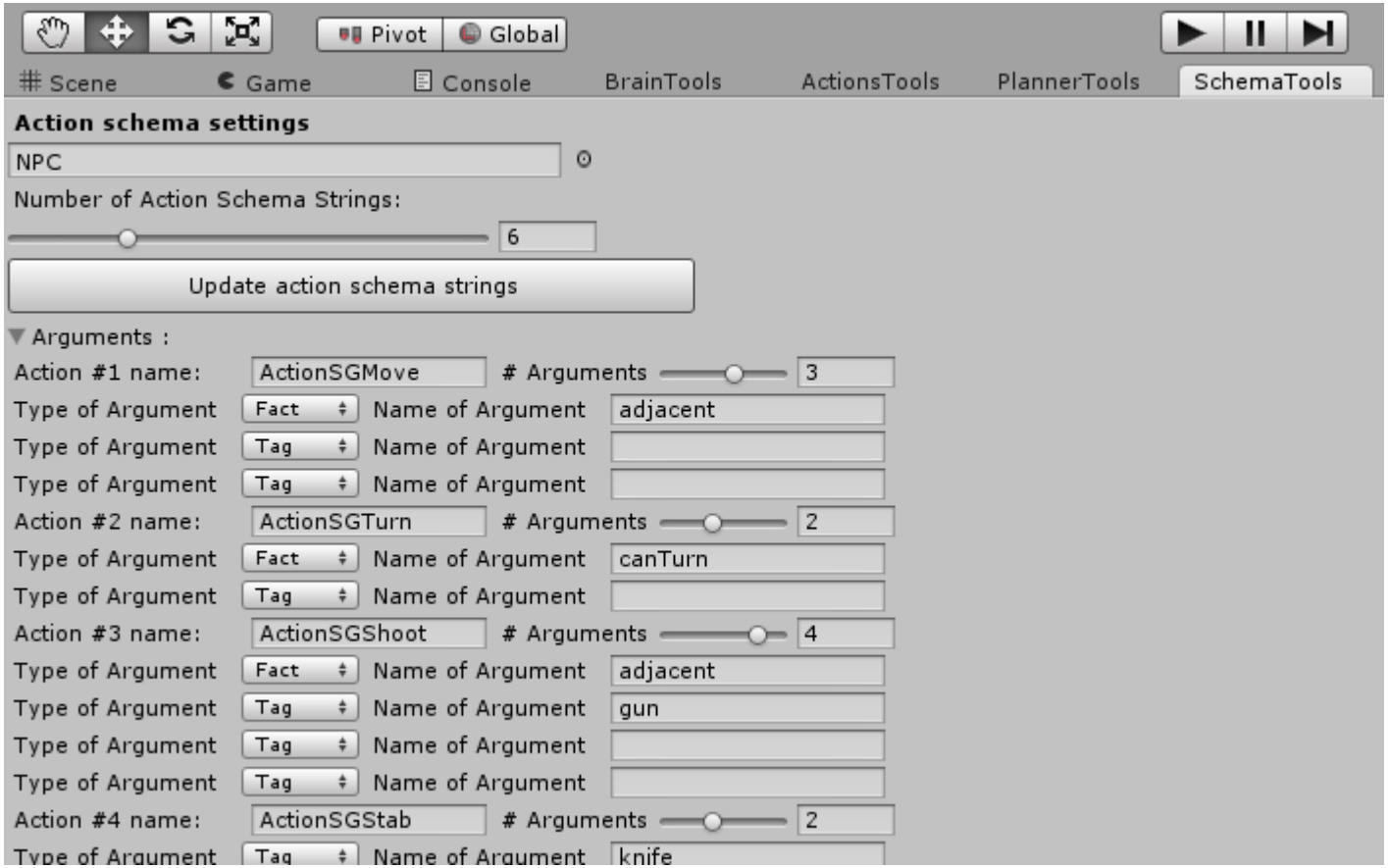
iThink: STRIPS planning in Unity3D

- Effects



iThink: STRIPS planning in Unity3D

- Filtering mechanism that uses Unity3D tags



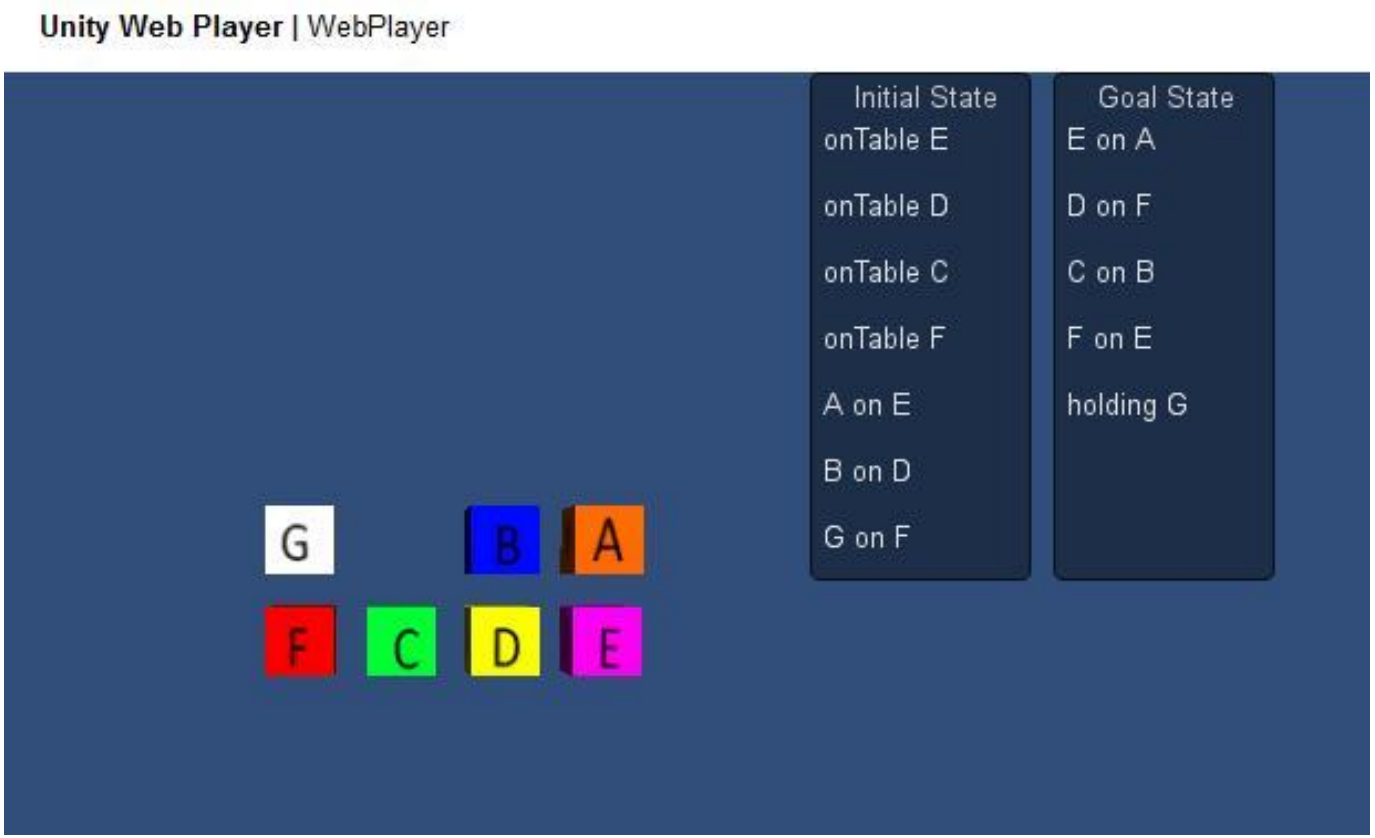
iThink: STRIPS planning in Unity3D

```
public class SimpleGameAgent : MonoBehaviour
{
   iThinkBrain brain ;
   public string [ ] schemaList = {
                          "ActionSGMove-3-Fact::adjacent",
                          "ActionSGTurn-2-Fact::canTurn",
                          "ActionSGShoot-4-Fact::adjacent-Tag::gun",
                          "ActionSGStab-2-Tag::location-Tag::knife",
                          "ActionSGPickUp-2-Tag::knife-Tag::location",
                          "ActionSGPickUp-2-Tag::gun-Tag::location“ };

   public void Awake() { //executed when NPC is constructed (GameObject
   is initialized)
      brain = new iThinkBrain() ;
      brain.ActionManager = new iThinkActionManager() ;
      ...
```

iThink: STRIPS planning in Unity3D

- Let's see a demo of the blocks world implemented in Unity3D with iThink!



SmartWorkersRTS in Unity3D

- B.Sc. project at the University of Athens
  - Ioannis Vlachopoulos
- Use iThink for STRIPS planning in a real-time strategy game to guide the actions of a worker unit
- In progress!

SmartWorkersRTS in Unity3D

- Real time strategy games (RTS) feature worker units that follow direct commands
  - Point and click
  - Get this resource, build this structure, etc
- The idea of this project is to allow some upgraded workers also take more long-term responsibilities
  - Requires a “rich” game-world where interesting interactions can take place between available resources, structures, objects

Experimenting with a commercial game

- Our game-world currently looks like this

Experimenting with a commercial game

- Our game-world currently looks like this

SmartWorkersRTS: Buildings

- Forest – Harvest or Hunt
- Gold Mine – Extract gold
- Armory – Get weapons and tools
- Farm – Get rice (ingredient for food-ration)
- Shop – Buy useful items
- Laboratory – Convert ingredients to other items (herbs → potions, gold → coins, rice → food-ration)
- Magic Tower – Provides spell scrolls (weapons)

SmartWorkersRTS: Objects

- Deer, Boar – Hunting
- Pick – Harvest tool
- Bow, Spear, Spell Scrolls – Hunting Weapons
- Food ration
- Potions
- Coins
- Gold – to produce coins
- Rice, raw meat – ingredients for food ration
- Herbs – Ingredients for potions

SmartWorkersRTS: Actions

- Buy an item from a shop
- Harvest an ingredient from forest
- Get a tool/weapon from armory
- Produce a new item in laboratory using some ingredients
- Hunt an animal with a weapon

SmartWorkersRTS: PDDL

- Shop ?x
- Laboratory ?x
- Natural-place ?x
- Building ?x
- Lives-in ?x ?y
- Money ?m (coins)
- Sells ?x ?y
- Is-converted-to ?x ?y
- Holding ?x
- Tool ?p
- Weapon ?x
- Provides ?x ?y

SmartWorkersRTS: PDDL

```
(:action get
 :parameters (?o ?from)
 :precondition (and (available ?from)
                    (building ?from)
                    (provides ?from ?o))
 :effect (holding ?o))
```

SmartWorkersRTS in Unity3D

- As different buildings are available at steps of the game, the worker can find different ways to achieve the same goals, e.g., bring food

- Interesting results arise when we consider different evaluation functions and search for the optimal solution
  - Use no coins
  - Prefer faster outcomes
  - …

SmartWorkersRTS in Unity3D

- Scenario 1
  - All buildings enabled
  - Default Cost Function – All actions cost 1
  - Goal : Holding(food-ration)

We expect that the agent will use only the shop to get a food-ration object, given that he is holding "coins".

SmartWorkersRTS in Unity3D

- Scenario 2
  - All buildings enabled
  - “money-saving” Cost Function – Buy action costs 7
  - Goal : holding(food-ration)

We expect that the agent will avoid using the shop, and either hunt or get resources from farm to produce a food-ration

SmartWorkersRTS in Unity3D

- Scenario 3
  - Shop and Farm disabled
  - Default Cost Function
  - Goal : holding(food-ration)

The agent has only hunting as the only means to produce a food-ration

SmartWorkersRTS in Unity3D

- Scenario 4
  - Shop and Laboratory disabled
  - Default Cost Function
  - Goal : holding(food-ration)

The agent will not find any plan

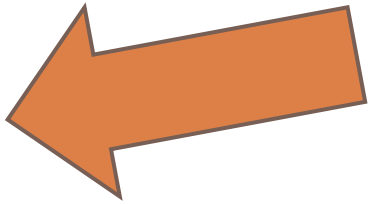
Experimenting with a commercial game

- Let's try a preliminary demo!

Experimenting with a commercial game

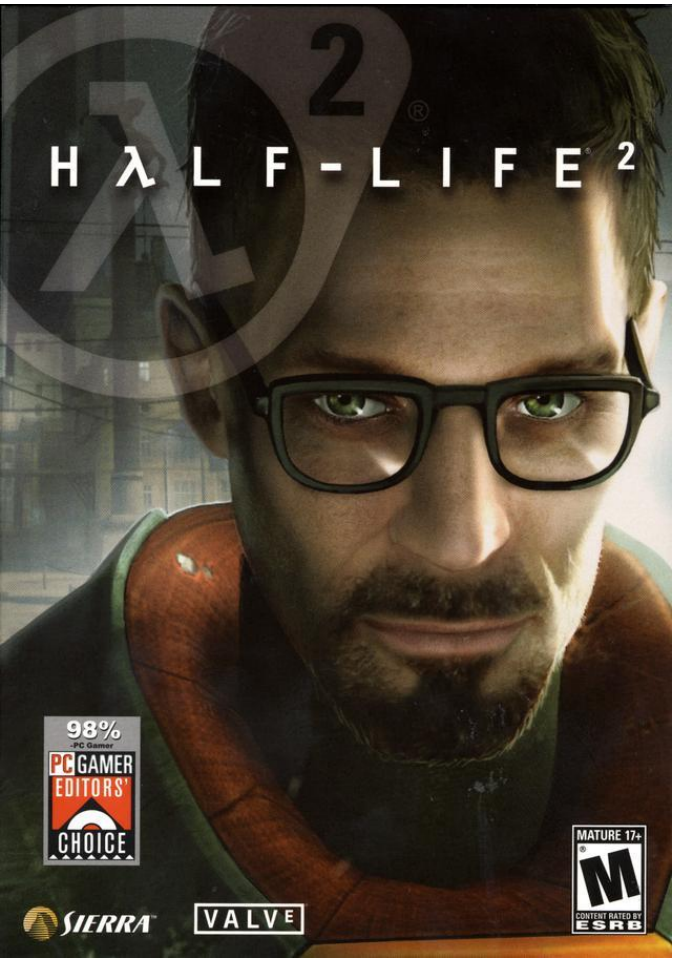
Experimenting with a commercial game

- Amazing tools available for (indie) game developers!



Experimenting with a commercial game

- Let's see some code from a commercial game

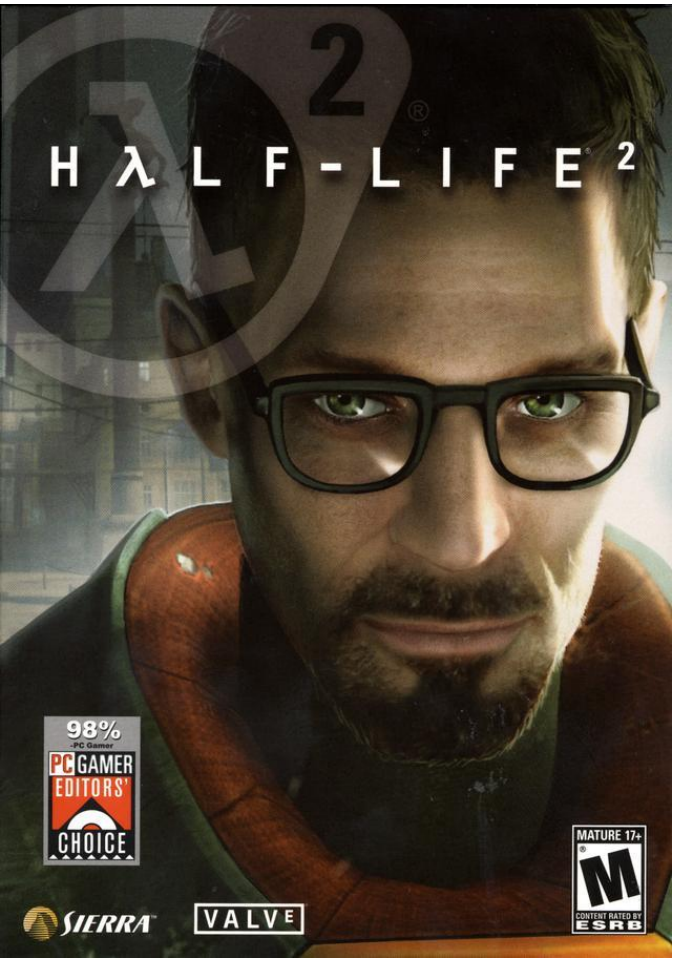


- HL2-SDK, npc_BaseZombie.cpp
- lines 1828-1870

```
switch ( m_NPCState )
{
case NPC_STATE_COMBAT:
…
case NPC_STATE_ALERT:
…
}
```

Experimenting with a commercial game

- Let's see some code from a commercial game



- Developers console
  - map
  - npc_create
  - npc_task_text
  - npc_route
  - npc_select
  - npc_tasks
  - npc_conditions
  - …

Artificial Intelligence and Video Games

- Source available!

Artificial Intelligence and Video Games

- Valve Developer Community, tools for Alien Swarm
  - https://developer.valvesoftware.com/wiki/Authoring_Tools/SDK_(Alien_Swarm)
- Programming overview
  - https://developer.valvesoftware.com/wiki/Category:Programming
- AI programming
  - https://developer.valvesoftware.com/wiki/AI_Programming

Next lecture

- Lecture 1: Game-inspired competitions for AI research, AI decision making for non-player characters in games
- Lecture 2: STRIPS planning, state-space search
- Lecture 3: Planning Domain Definition Language (PDDL), using an award winning planner to solve Sokoban
- Lecture 4: Planning graphs, domain independent heuristics for STRIPS planning
- Lecture 5: Employing STRIPS planning in games: SimpleFPS, iThinkUnity3D, SmartWorkersRTS
- Lecture 6: Planning beyond STRIPS

Bibliography

- References
  - iThink: A Library for Classical Planning in Video-games. Vassileios-Marios Anastassiou, Panagiotis Diamantopoulos, Vassos Stavros, Manolis Koubarakis. In Proceedings of the 7th Hellenic Conference on Artificial Intelligence (SETN), 2012.
  - Real-time Action Planing with Preconditions and Effects. Stavros Vassos. Game Coder Magazine, March 2012.