Universitat de Girona
Escola Politècnica Superior

Snowman Editor

A level design tool for the game A Good Snowman Is Hard To Build

Study: Grau en Enginyeria Informàtica

Document: Summary

Author: Gerard Martin Teixidor

Email: gerardmartinteixidor@gmail.com

Director: Gustavo Ariel Patow and Mateu Villaret Auselle

Department: Informàtica, Matemàtica Aplicada i Estadística (IMAE)

Area: Llenguatges i Sistemes Informàtics (LSI)

Call: September/2018

Chapter 1

Introduction

In the design of video-games, one of the most important aspects to consider is the difficulty in the design of the different levels. A common problem is the possibility of finding possible solutions that were not planned. These solutions may not be desired because they can break the game dynamics. Hence, it is introduced the idea to create a tool to facilitate the design of those levels by detecting unwanted solutions.

This problem can be seen as a Planning problem, since, given the game state and the possible actions the player can perform, the solutions a level may have can be searched. Once the solutions are found, those that were not considered in the original design can be removed.

Usually this unwanted solutions are shorter, therefore easier solutions, than the designed solution. One of the goals of this project is, given a level, find the optimal solution, which will allow to check if there exist any solution easier than the designed one. Finding an optimal solution is not a simple task since exploring a game search tree by uninformed algorithms can be impossible in a reasonable time. Hence, one part of this project will be focused on reducing this previously described problem to Satisfiability Modulo Theories (*SMT*).

Since finding the optimal is a hard problem, in this project, two different problem relaxations will be presented which will allow to solve the problem in a plausible time.

1.1 Personal motivations

I had chosen this project because it merges the two worlds which I am more passionate about, video-games and optimization. Furthermore, is also a great way to go in depth about AI, since Planning is a well known branch of this topic.

This project has was not easy, despite this I enjoyed working on it and spending much free time on it.

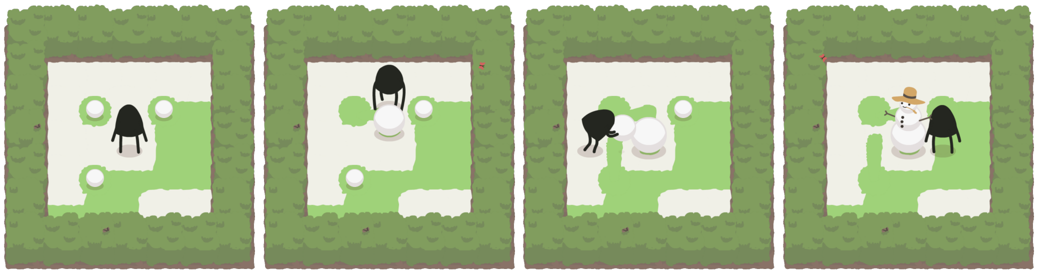
1.2 Project objectives

The main objectives of this project is the development of this tool for a concrete game, *A Good Snowman is Hard to Build*. This tool, named *Snowman Editor*, will consist of the following parts:

- Solve a given level, with an heuristic approach, using the Planning Domain Definition Language (*PDDL*) and multiple solvers
- Solve a given level, with an optimal approach, using Satisfiability Modulo Theories (*SMT*)
- A graphical level editor

1.3 A Good Snowman is Hard to Build

A Good Snowman is Hard to Build is a commercial game based on the popular Sokoban transport puzzle. The goal is to build a snowman from tree balls, each one smaller than the other.



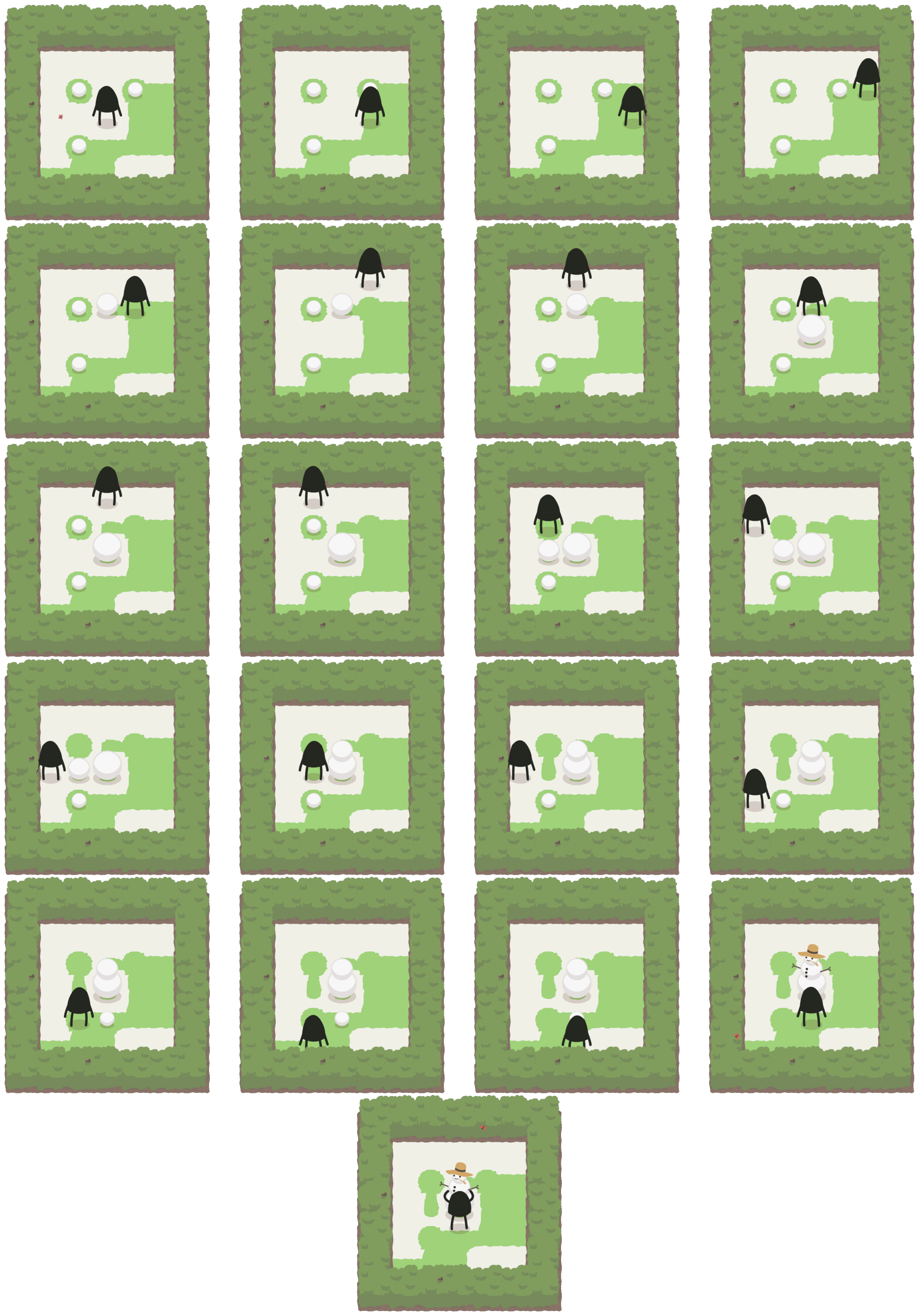
The game consists of multiple levels arranged in an open world layout. For this project each individual level will be seen as a different problem instance. A complete instance solution can be found in the Example 1.3.1.

1.3.1 Game mechanics

The game has two worlds with different mechanics each one. This project is focused on the “real world“ mechanics but, with minor changes, it should be easy to adapt it to the “dream world“ mechanics. The rules describing the “real world“ mechanics are the following:

1. The character can only push the balls from behind.
2. The character can also push and pop a ball over other balls.
3. A ball can not be pop from a ball to directly be on top of another ball.
4. The ball below must be larger.
5. Each ball can have three sizes: small, medium and large.
6. When the ball is rolled on top of snow on the floor it increases its size by one unit, and removes the snow on the floor.
7. When rolled, a large ball does not increase its size though it also removes the snow.
8. Balls can only grow.
9. A level can have multiple snowmans. The number of balls in a level has to be a multiple of three and bigger than 0.
10. Once the snowman is built, you can not pop the top ball. This rule currently is not implemented in the SMT approach.
11. A snowman is formed by three balls.
12. A snowman can be built anywhere.

Example 1.3.1. *A complete solution to the Andy level:*



Chapter 2

Design

2.1 Snowman editor

The *Snowman editor* (Figure 2.1) is a complete tool which facilitates the level design by allowing the user to create a new level, try it in-game and also find a solution. The tool has multiple solvers which use the SMT approach but also allows to generate PDDL problem instances which can be used externally with the desired planner.

2.2 PDDL Approach

The PDDL approach aims to solve the *Snowman problem* by using the PDDL standard Artificial Intelligence planning language. Because of the limitations with PDDL and its planers, two different approaches have been taken.

2.2.1 *adl* approach

The main idea behind the *adl* approach is to associate a distinct object to each possible locations of the level at hand. But, the main problem with this approach is the number of action which because it is to large, makes the search impossible.

2.2.2 object-fluents approach

The *object-fluents* approach aims to reduce the number of actions, but, in the current state of the art, there do not exist planners which supports this newer language subset.

2.3 SMT Approach

The SMT approach allows to find an optimal solution to the *Snowman problem.* This is desired since the main objective of the project is to find undesired solutions, which usually are easier solutions that the wanted solution.

During the encoding development process three different encodings have been created since each one has its advantages an inconveniences. The theory used is Linear Integer Arithmetic since integer variables and linear arithmetic expressions are required.

2.3.1 Basic encoding

A first approach to encode the *Snowman problem* is to encode all possible actions a player can do in the game. Even though there are only four actions in the game (up, down, left, right)[1], since they have different effects, these actions can be split into two categories: character movement actions and ball movement actions.

[1]The game is played using the keyboard arrows

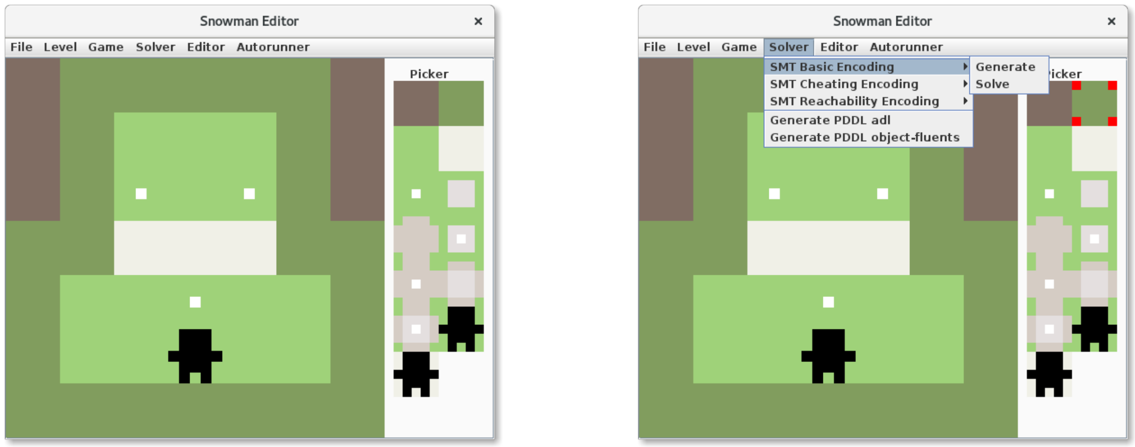

Snowman Editor
File Level Game Solver Editor Autorunner
Picker
Snowman Editor
File Level Game Solver Editor Autorunner
SMT Basic Encoding
SMT Cheating Encoding
SMT Reachability Encoding
Generate PDDL adl
Generate PDDL object-fluents
Generate
Solve

Figure 2.1: *Snowman Editor* with the level Sarah loaded. The tool is composed by a level representation (right), a tile type picker (right) and a menu bar (top) which allow to use all the tool functionalities.

This way, this first encoding encodes four actions for the character movement actions and four actions for the ball movement actions, but for each ball. To perform a ball movement action, first the player has to travel next to the ball using the character movements actions.

The preliminary result however, showed that this first encoding was not able to solve many levels in a reasonable time.

2.3.2 Cheating encoding

This encoding is based on the previous encoding, but it removes the character movement actions allowing the character to teleport anywhere on the map as long as is not occupied. This can be seen as cheating, as a ball can block a path to a candidate ball movement action.

2.3.3 Reachability encoding

The *Reachability encoding* is based on the previous *Cheating encoding*, but it introduces a new reachability precondition. These preconditions ensure that there is a path from the position of the character when doing a ball movement action to the position of the character to perform the next ball movement action.

Reachability precondition

The reachability precondition encodes an *s-t-reachability* problem extending the *Topological Sort with Indices* described in the article *SAT modulo graphs: Acyclicity*. Is important to remark that in contrast to the article, this extension must deal with non-static graphs in the sense that it is not know, when the the encoding is being build, which will be the source and which will be the target of the s-t-reachability problem. Neither is not known which positions will be occupied. Therefore, we believe that this is a remarkable contribution of this project.