COMP3702 Artificial Intelligence (Semester 2, 2023)
The DragonGame AI Environment

This document provides a high-level description of the DragonGame environment. The support code can be found at: `https://gitlab.com/3702-2023/a1-support`.

Background

"Untitled Dragon Game"[1] or simply DragonGame, is a 2.5D Platformer game in which the player must collect all of the gems in each level and reach the exit portal, making use of a jump-and-glide movement mechanic, and avoiding landing on lava tiles. DragonGame is inspired by the "Spyro the Dragon" game series from the original PlayStation.

To optimally solve a level, your AI agent must find a sequence of actions which collects all gems and reaches the exit while incurring the minimum possible action cost.

Levels in DragonGame are composed of a 2D grid of tiles, where each tile contains a character representing the tile type. An example game level is shown below:

```
XXXXXXXXXXXXXXXXXXXX
XX             XXXXXX
X G ===       ===XXXX
XXX=X==       =====XX
X  ===       G  ====X
X  ==       XX==  ==X
X  == G      ===   =X
X=XXXXXX  = ===   =X
X===  X   = ===  G=X
X===  X  ==  =  XXXX
X===     ==X = XX  X
X=XX G   =XX =    EX
X=PXXX  XXXXXXX   =X
XXXXXX**XXXXXXX**XXX
```

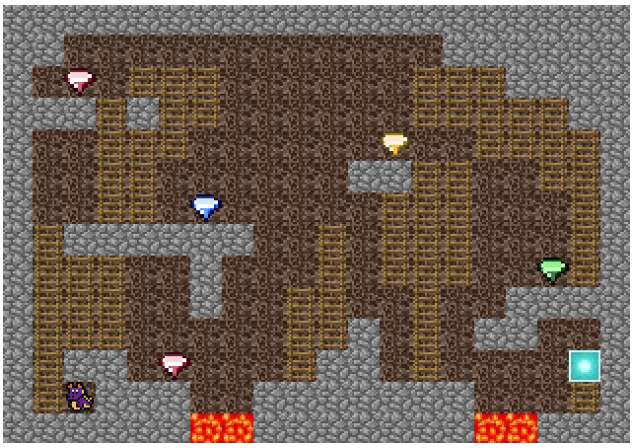


Figure 1: Example game level of DragonGame, showing character-based and GUI visualiser representations

[1] This full game title was inspired by Untitled Goose Game, an Indie game developed by Australian developers in 2019

Game state representation

Each game state is represented as a character array, representing the tile types and their position on the board. In the visualizer and interactive sessions, the tile descriptions are graphical assets, whereas in the input file these are single characters.

Levels can contain the tile types described in Table 1.

Table 1: Table of tiles in DragonGame, their corresponding symbol and effect

Tile	Symbol in Input File	Image in Visualiser	Effect
Solid	'X'		The player cannot move into a Solid tile. Walk and jump actions are valid when the player is directly above a Solid tile.
Ladder	'='		The player can move through Ladder tiles. Walk, jump, glide and drop actions are all valid when the player is directly above a Ladder tile.
Air	' '		The player can move through Air tiles. Glide and drop actions are all valid when the player is directly above an Air tile.
Lava	'*'		The player cannot move into a Lava tile. Moving into a tile directly above a Lava tile results in Game Over.
Gem	'G'		Gems are collected (disappearing from view) when the player moves onto the tile containing the gem. The player must collect all gems in order to complete the level. Gem tiles behave as 'Air' tiles, and become 'Air' tiles after the gem is collected.
Exit	'E'		Moving to the Exit tile after collecting all gems completes the level. Exit tiles behave as 'Air' tiles.
Player	'P'		The player starts at the position in the input file where this tile occurs. The player always starts on an 'Air' tile.

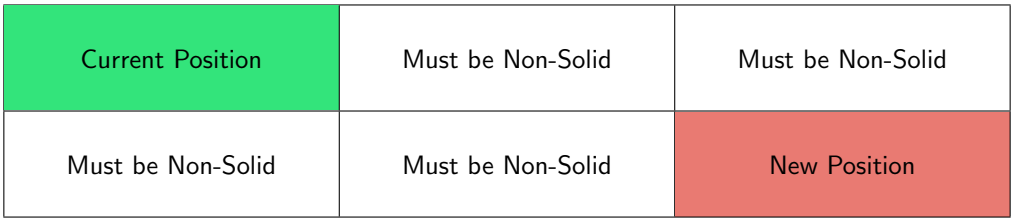
Actions

At each time step, the player is prompted to select an action. Each action has an associated cost, representing the amount of energy used by performing that action. Each action also has requirements which must be satisfied by the current state in order for the action to be valid. The set of available actions, costs and requirements for each action are shown in Table 2.

Table 2: Table of available actions, costs and requirements

Action	Symbol	Cost	Description	Validity Requirements
Walk Left	wl	1.0	Move left by 1 position	Current player must be above a Solid or Ladder tile, and new player position must not be a Solid tile.
Walk Right	wr	1.0	Move right by 1 position	
Jump	j	2.0	Move up by 1 position.	
Glide Left 1	gl1	0.7	Move left by 1 and down by 1.	Current player must be above a Ladder or Air tile, and all tiles in the axis aligned rectangle enclosing both the current position and new position must be non-solid (i.e. Air or Ladder tile). See example below.
Glide Left 2	gl2	1.0	Move left by 2 and down by 1	
Glide Left 3	gl3	1.2	Move left by 3 and down by 1	
Glide Right 1	gr1	0.7	Move right by 1 and down by 1	
Glide Right 2	gr2	1.0	Move right by 2 and down by 1	
Glide Right 3	gr3	1.2	Move right by 3 and down by 1	
Drop 1	d1	0.3	Move down by 1	Current player must be above a Ladder or air tile, and all cells in the line between the current position and new position must be non-solid (i.e. Air or Ladder tile).
Drop 2	d2	0.4	Move down by 2	
Drop 3	d3	0.5	Move down by 3	

Example of glide action validity requirements for `GLIDE_RIGHT_2` ('gr2'):



Interactive mode

A good way to gain an understanding of the game is to play it. You can play the game to get a feel for how it works by launching an interactive game session from the terminal with the following command:

```
$ python play_game.py <input_file>.txt
```

where `<input_file>.txt` is a valid testcase file from the support code with path relative to the current directory, e.g. `testcases/L1.txt`

In interactive mode, type the symbol for your chosen action (e.g. '`wl`') and press enter to perform the action. Type '`q`' and press enter to quit the game.

Your assignment task

Your task is to develop a program that outputs a path (series of actions) for the agent (i.e. the Dragon) utilising UCS and A* search, and to provide a written report explaining your design decisions and analysing your algorithms' performance. The details of the assignment task will be provided in a separate assignment description document.