Platformata – Game Design with Automata Theory

Ahmed Shuaib

May 28, 2024

Introduction

Automata Theory is the study of abstract machines. With the help of this branch of theoretical computer science, many methods and tools have been developed that are widely used to analyze systems, design compilers, and programming languages. As software continues to grow in complexity many modern development tools are developed to optimize the development experience. This paper aims to explore the use of methods and tools of automata to achieve an optimized development experience in the context of 2D game design and outline that can be modified and/or built upon.

Literature Review

There are many ways of incorporating automata in game design that have been considered. Many design decisions take place during the development of a game, these decisions can vary based on the genre of the game.

When developing 2D games that fall in the genre of role-playing games (RPGs), adventure games, and puzzles. A procedural-level generation has been demonstrated using cellular automata. In a thesis from 2022, A genetic algorithm is used to evolve the cellular automata rules applied to generate game levels. It is important to note that a procedural content generator designed using this method is meant to be used during the game development process rather than at runtime. The resulting game levels can have a significant impact on a game's replayability, engagement, and scalability. [1]

Games designed solely using automata are few, but they do exist. Most games designed using automata are finite, however, in a paper from 2016, students designed an infinite runner game using mealy machines. The game consisted of several states: running, jumping, flying, and game over. The paper concluded that games designed using automata are less prone to bugs and the development process is simplified by handling many implementation decisions in the design phase. [2]

An example of a game designed solely using automata is Conway's Game of Life. Developed by John Conway the Game of Life is a "zero-player" game utilizing cellular automata. The game is essentially a square grid containing cells that evolve based on three rules: birth, death, and survival. A dead cell will become alive in the next "time step" if three of its eight neighboring cells are alive. A cell can die in the next time step due to overcrowding, which occurs when it has four or more alive neighbors, or from exposure, which happens when it has one or fewer alive neighbors. A cell survives the current "time step" if the cell has either two or three alive neighbors. [3]

Game Description

A game design for a fairly common 2D game genre is demonstrated using non-deterministic finite automata, a 2D Platformer, named Platfomata. As the player of this 2D platformer, your goal is to reach the end most efficiently. In as few attempts as possible, and as fast as possible. At the start of each level, the player will be given some time. As you move around the level each state you encounter may take time, the amount of time some state takes generally varies based on the type of state, however, in some special cases, two states of the same type may take different amounts of time. When the player runs out of time, they transition back to the initial time state for another attempt. Most levels will require multiple attempts to reach the end. Any interactions the player has with the level are maintained if the allotted time is depleted. As the levels go on, the complexity of each level increases and the player will need to try various paths and implement strategies to reach the end in the given time. There are 4 levels, a detailed breakdown of the levels is provided below.

Level 1

The player is provided with 36 seconds. The level consists of multiple platforms and stairs, jump pads as well as buttons and ghost platforms. The level will take around 2 attempts to complete and requires the player to pursue different paths in each attempt.

Level 2

The player is provided with 120 seconds. The level consists of multiple platforms and stairs, jump pads as well as buttons ghost platforms. This level also introduces ghost walls and cubes and will take around 2 attempts to complete, requiring the player to pursue different paths in each attempt.

Level 3

The player is provided with 90 seconds. The level consists of multiple platforms and stairs, jump pads as well as buttons ghost walls, and platforms. This level also introduces platform walls. This level will take around 2-3 attempts based on the paths the player pursues.

Level 4

The player is provided with 90 seconds. The level consists of multiple platforms and stairs, jump pads as well as buttons, ghost walls, platforms, and platform walls. This level will take around 2-3 attempts based on the paths the player pursues.

Game Design

It is important to note that this 2D platformer is designed in a way that acknowledges many decisions that take place during the development of a game. Many design decisions that simplify the development process. Aspects such as scalability and modularity of the game have been considered. That is to say if one intends to follow the game design outlined below, it will be relatively simple to implement the logic of this platformer and build on it using modern development tools. Designing game mechanics that rely on a time-distance/time-state relationship, adding additional states, designing complex levels, and even utilizing procedural generation, the possibilities are vast. A detailed breakdown of the game design is provided below.

Input Alphabet

$\sum_{\text{Move}} = \{ \text{U, D, L, R} \}$

$\sum_{\text{Action}} = \{ \text{P, J, O} \}$

$\sum_{\text{Game}} = \{ \text{T}_\text{n},\ \text{TA}_\text{n},\ \text{A}_\text{n},\ \text{B}_\text{n},\ \text{NB}_\text{n},\ \text{D}_\text{n},\ \text{ND}_\text{n} \}$

Description of input alphabet

$\sum_{\text{Move}}$: Defines the movement of the player. Up (U), Down (D), Left (L), and Right (R).

$\sum_{\text{Action}}$: Defines the actions the player can take. Push button (P), Jump (J), Open door (O).

$\sum_{\text{Game}}$: Defines some data for the game level. Counters like timeout when $\text{T}_\text{n} \leq 0$, total attempts TA_n, and attempts at each state A_n. These can be represented by some integer value. Status like button status B_n and NB_n, and status of state D_n and ND_n. These can be represented by some boolean value (0/1). This data is reset at the start of each level. It is important to note that the counters that keep track of each state use the state's n value for identification, this means that if more than one state has the same n value the counter will track all those states. If one wishes to individually track the attempts at some state, that state must have a unique n value. Another behavior to note is that the value of any status prefixed with 'N' will be the inverse of its non-prefixed counterpart, this behavior, however, is up to the designer. The default values of the data are as follows:

$\text{T}_\text{n} = 0 \quad \text{TA}_\text{n} = 0 \quad \text{A}_\text{n} = 0 \quad \text{B}_\text{n} = 0 \quad \text{NB}_\text{n} = 1$
$\text{D}_\text{n} = 0 \quad \text{ND}_\text{n} = 1$

States of the game

Time State ($\mathbf{T_n^t}$)

Gives the player time and tracks total attempts.
Effects: $\text{T}_\text{n} = \text{t} \quad \texttt{++}\text{TA}_\text{n}$

Platform State ($\mathbf{P_n^t}$)

Platforms that reduce time.
Effects: T_n `-=` $\text{t} \quad \texttt{++}\text{A}_\text{n}$

Stair State ($\mathbf{S_n^t}$)

Stairs that reduce time.
Effects: T_n `-=` $\text{t} \quad \texttt{++}\text{A}_\text{n}$

Button State ($\mathbf{B_n^t}$)

Buttons that reduce time and set the status of buttons.
Effects: T_n `-=` $\text{t} \quad \text{B}_\text{n} = 1 \quad \text{NB}_\text{n} = 0 \quad \texttt{++}\text{A}_\text{n}$

Button State ($\mathbf{NB_n^t}$)

Buttons that reduce time and set the status of buttons.
Effects: T_n `-=` t $B_n = 0$ $NB_n = 1$ `++`A_n

Jump State ($\mathbf{J_n^t}$)

Jump pads that reduce time.
Effects: T_n `-=` t `++`A_n

Hole state ($\mathbf{H_n}$)

Innocent Holes.
Effect: `++`A_n

Ghost Platform State ($\mathbf{GP_n^t}$)

Platforms that reduce time and depend on the status of buttons.
Effects: T_n `-=` t `++`A_n

Ghost Wall State ($\mathbf{GW_n^t}$)

Walls that reduce time and depend on the status of buttons.
Effects: T_n `-=` t `++`A_n

Platform Wall State ($\mathbf{PW_n^{h^d}}$)

Walls that reduce time-based on damage and when disabled functions like platforms.

Effects: T_n `-=` $(h - (d \cdot (\texttt{++}A_n)))$

$D_n = (h - (d \cdot A_n)) \leq 0$

$ND_n = (h - (d \cdot A_n)) > 0$

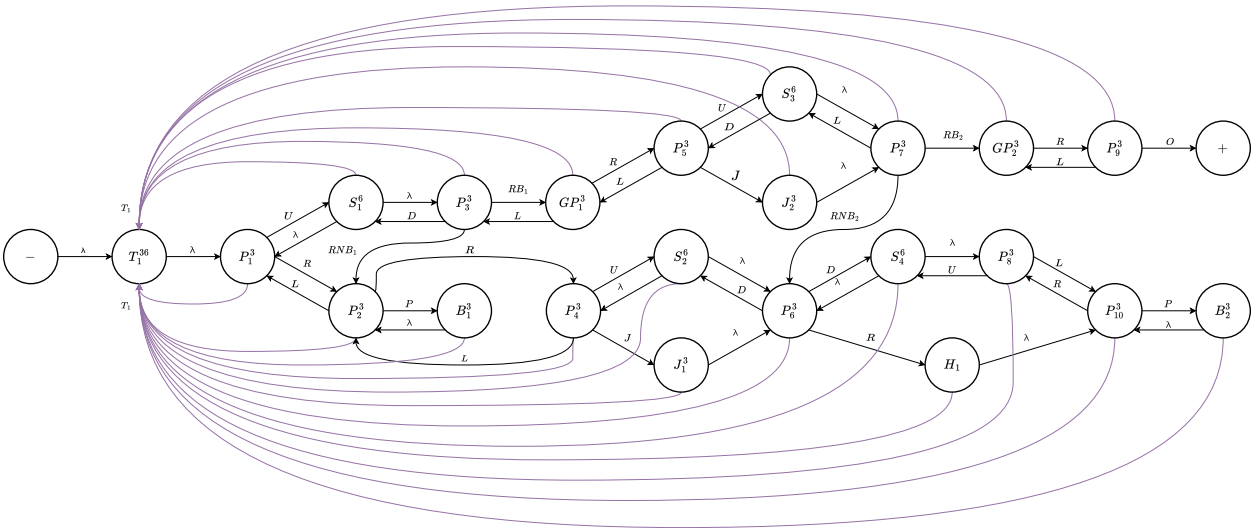
Cube state ($\mathbf{C_n^{h^d}}$)

Cubes that reduce time based on damage and when disabled allow the player to move forward.

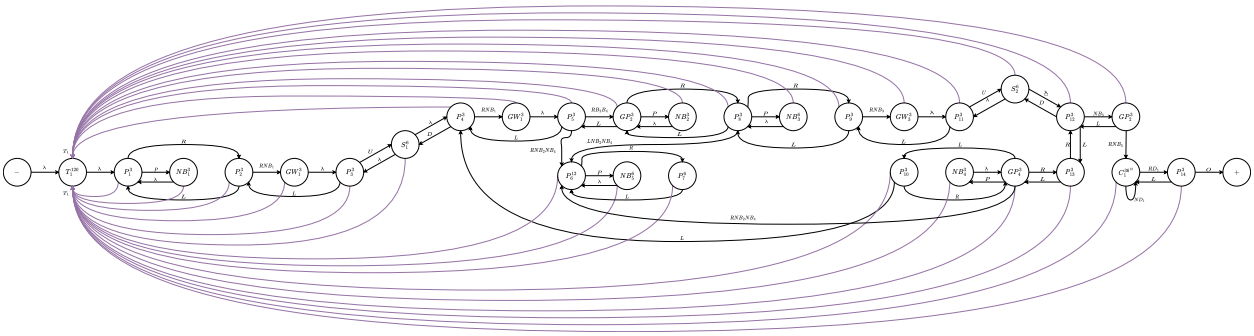
Effects: T_n `-=` $(h - (d \cdot (\texttt{++}A_n)))$

$D_n = (h - (d \cdot A_n)) \leq 0$

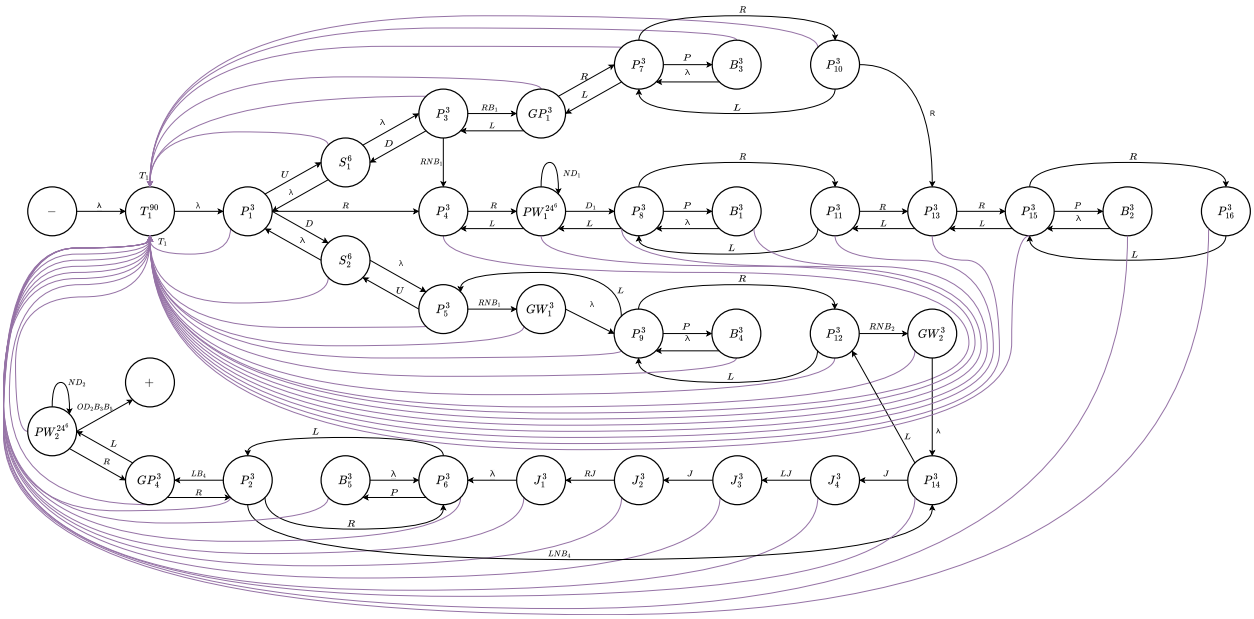
$ND_n = (h - (d \cdot A_n)) > 0$



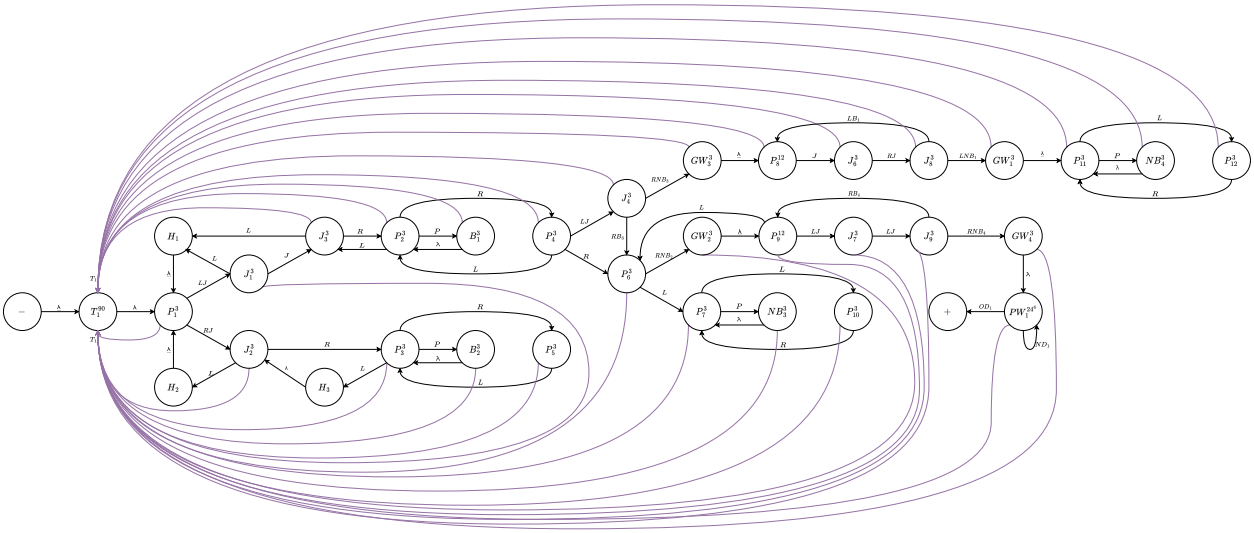
Level 1



Level 2



Level 3



Level 4

Conclusion

Incorporating automata into game design offers a versatile approach. Allowing design decisions tailored to each game's genre. Offering a pleasant gameplay experience but also a streamlined design and development process. Suffice it to say using methods and tools of automata is a worthwhile approach to achieve an optimized development experience for game developers.

References

[1] Adel Sabanovic and Amir Khodabakhshi, Evolved cellular automata for 2D video game level generation, 2022.

[2] Abid Jamil, Engr. AsadUllah and Mohsin Rehman, An Infinite Runner Game Design using Automata Theory, 2016.

[3] Kuldeep Vayadande, Ritesh Pokarne, Tanmay Patil, Mahalakshmi Phaldesai, Prachi Kumar, and Tanushri Bhuruk, Simulation of Conway's Game of Life using cellular automata, 2022.