Teoría de autómatas

en el diseño de juegos

Piero Alexis Violeta Estrella

Enero, 2022

Universidad Nacional de Ingeniería

Tabla de contenidos i

Tabla de contenidos ii

Introducción

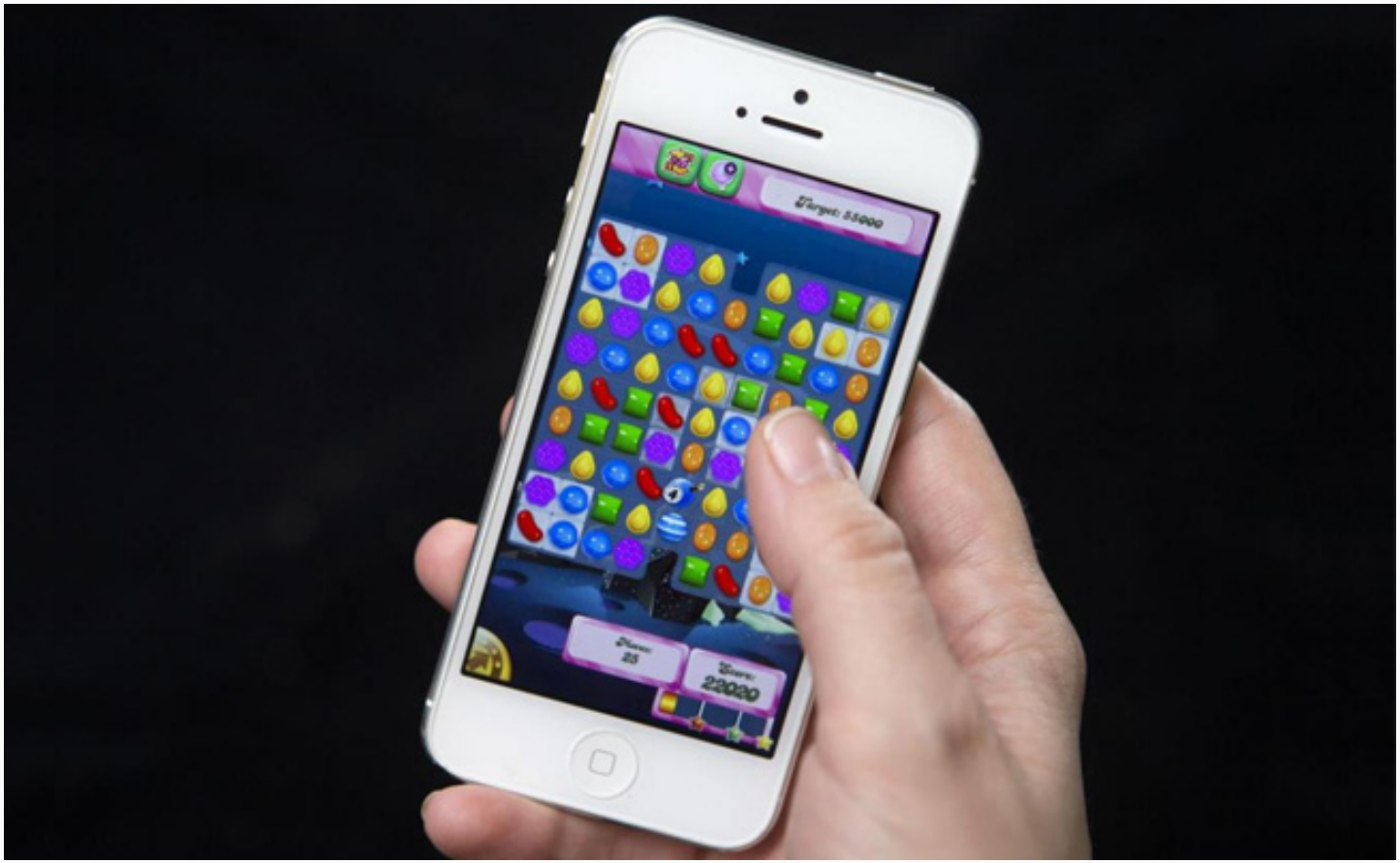
Motivación

Desde sus inicios, los videojuegos han sido una forma de entretenimiento que no pasaban desapercibida.

Salón de videojuegos Arcade en Japón

Motivación

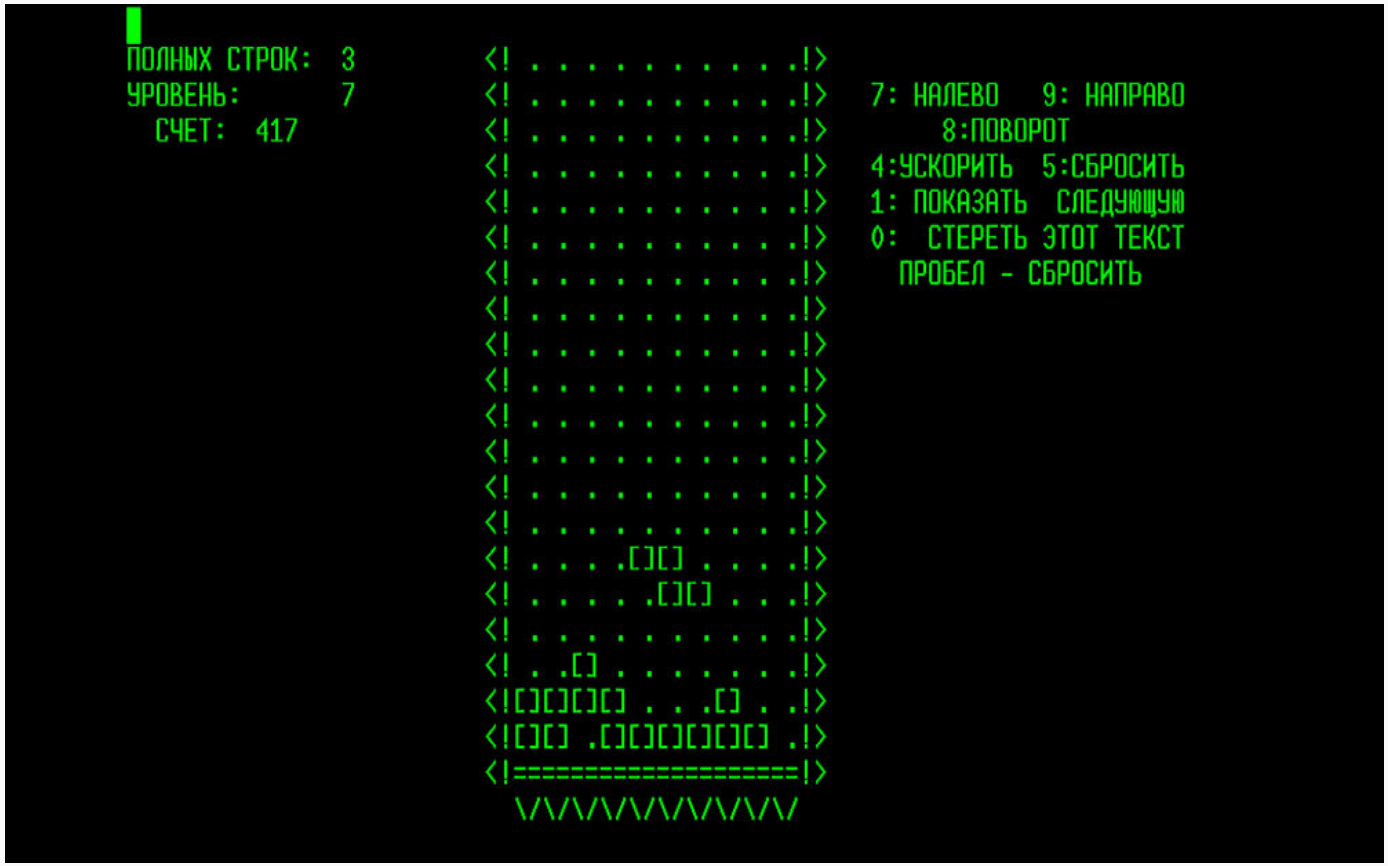
Desde salones Arcade, con grandes máquinas, hasta juegos casuales de hoy en día, que pueden ser disfrutados desde un teléfono celular.



Videojuego casual en teléfono móvil

Motivación

El desarrollo de un videojuego puede ser desarrollado por una sola persona.



Tetris (1984)

Motivación

También puede ser desarrollado por un equipo entero.

Half Life (1998)

Motivación

Un equipo entero de desarrollo de un videojuego de esa magnitud puede incluir:

- Modeladores
- Programadores
- Artistas
- Diseñadores de juegos
- Músicos
- Escritores

Solo mencionando algunos puestos es fácil notar que son muchas personas involucradas.

Motivación

Un problema común entre personas programadoras y no programadoras es la comunicación. Más en especifico, el problema de entendimiento entre programadores y diseñadores de juegos.

Motivación

Change Weapons	Mouse wheel	Y button
Sniper Zoom		Left Trigger
Display Goals/Pause		Start Button

The Interface Commandments

In keeping with the design goals of BioShock, we've eliminated or radically streamlined almost all of the complexity of interface management. The shooter market simply will not bear overly complex interfaces.

We've accomplished this via a couple of key changes, which we like to call the "Three Commandments"

1 - Thou shall not have to deal with an overly complex inventory system

In System Shock 2, the player had to deal with numerous inventory items that he dragged around the screen with a mouse and placed in various inventory "slots". Not the case in BioShock. The three types of objects the player can pick up are:

- Ammo and Resources - Ammo is picked up and stored like every other FPS on the planet. As the player walks over it in the world, he "vacuums" it up. The player can never "drop" ammunition, he can only expend it via his weapons. Resources such as nanites, the currency of the BioShock world, are handled in the same way except they are expended in the tech stations (see below).
- Weapons - Weapon pick up and weapon dropping is handled much like Halo. The player can only have four weapons at any one time.
- Quest Items - The player will automatically pick up quest items and automatically "use" them at the appropriate time (for example, if the item is a keycard, it will be used when the player interacts with the relevant locked door. Or if the player wants to upgrade his crustacean form, first he must find a genetic sequence that will allow him to do so. These items will be automatically expended when the player interacts with the appropriate genetic modification machine.

2 - Thou shall not be bogged down with countless interface screens

System Shock 2 featured a bewildering variety of Windows-like interface screens that the player had to manage-- Weapon modding, hacking, paper doll, inventory, research, weapon repair, etc. The player was forced to manage this complexity through a Windows 95 style set of interface screens, full of buttons, tabs, sliders and other widgets.

One exception to this rule was character growth, which was handled in a unique, refreshing manner. The player couldn't level up his character at will. He could only do at specific "Skill Machines" located around the environment.

In Bioshock, we take this system and run with it. There will be numerous machines in the world for the player to interact with. The interface for using all of these machines is identical: i.e. you press a button. Let's use an example:

Wandering through the installation, the player comes across a machine mounted on the wall. From the distance, it looks something like this:

This is a temperature control device. So what, you may ask? Well, the creatures in BioShock (and the player) are often temperature sensitive. Heating things up around the a hydrozoan mutation can slow them down, distract them and make them less likely to notice you coming up behind them. De-humidifying the air will make many foes weak and dehydrated.

The red thermometer icon indicates that the temperature as at the hottest of it's 3 (perhaps 5) possible settings. The player can tell this from across the room. As the player approaches the machine, they hit the RMB, or the B button on the Xbox. This image zooms up in their screen (it will be a 3d model in the world, not a full screen interface)

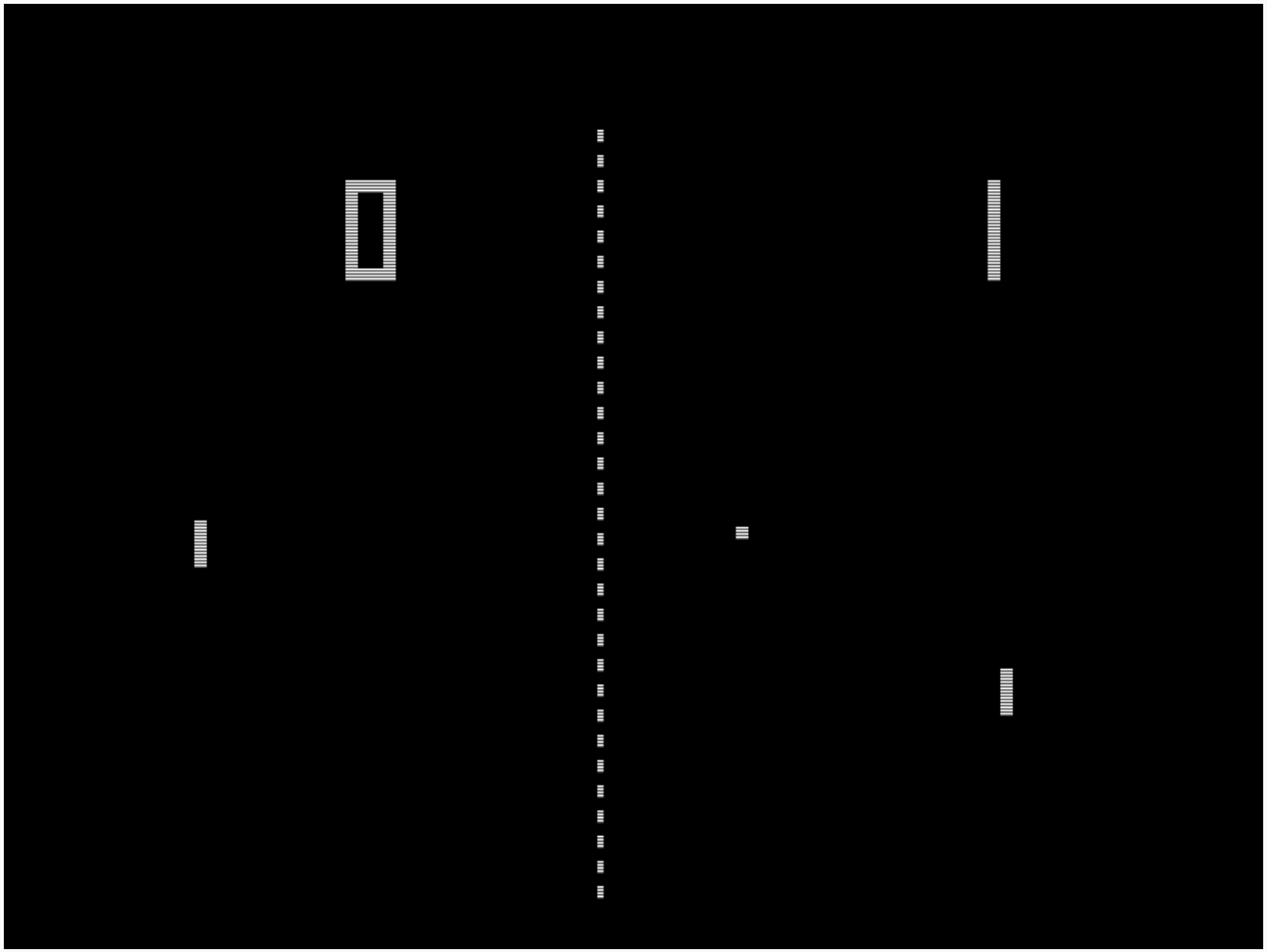
The player can then use the forward and back keys (W and S) on the pc or the left thumpad on the Xbox to adjust the temperature. The nanite costs indicates how many nanites it will take to make this adjustment. The bar sinister on the top arrow means that the temperature is as hot as the system will allow. When the player right clicks or hit the B button again, this interface is dismissed and any temperature adjustments are made.

The Bioshock game design document by Irrational Games

Problema

Planteamiento del problema

Juego de reglas simples



Pong (1972)

Planteamiento del problema

Juego de reglas complejas

God Of War (2018)

Algunos enfoques

- Computing Game Design with Automata Theory
- Roller Coaster Game Design using Automata Theory
- An Infinite Runner Game Design using Automata Theory
- Re-designing the Pacman Game using Push Down Automata

Hipótesis

Hipótesis General

Es posible generar un marco teórico sobre el diseño de juegos basado en autómatas.

Hipótesis Especificas

- La teoría de autómatas trae un beneficio al entendimiento por parte del programador de reglas y/o mecánicas diseñadas por el diseñador de juegos.
- El uso de un tipo de autómata adecuado para determinados tipos de videojuegos puede facilitar su implementación en gran medida.

Conceptos previos

Autómata Finito Determinista

Un Autómata Finito Determinista(AFD) A es una 5-upla $(Q, \Sigma, \delta, q_0, F)$, donde:

- Q denota un conjunto finito de estados.
- Σ denota el alfabeto de entrada.
- δ denota una función de transición, que toma como entrada un estado y un carácter de entrada, y retorna un estado.
  Formalmente: $\delta : Q \times \Sigma \to Q$
- q_0 denota el estado inicial, donde $q_0 \in Q$
- F denota un conjunto finito de estados de aceptación.

Autómata Finito No Determinista

Un Autómata Finito No Determinista(AFND) B es una 5-upla $(Q, \Sigma, \delta, q_0, F)$, donde:

- Q denota un conjunto finito de estados.
- Σ denota el alfabeto de entrada.
- δ denota una función de transición, que toma como entrada un estado y un carácter de entrada, y retorna un subconjunto de Q. Formalmente: $\delta\colon Q \times \Sigma \to \mathcal{P}(Q)$
- q_0 denota el estado inicial, donde $q_0 \in Q$
- F denota un conjunto finito de estados de aceptación.

Autómata de pila

Un Autómata de pila C es una 7-upla $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, donde:

- Q denota un conjunto finito de estados.
- Σ denota el alfabeto de entrada.
- Γ denota el alfabeto de pila.
- δ denota una función de transición, que toma como entrada un estado, un carácter de entrada(incluso cadena la vacía ε) y un carácter de la pila. Este a su vez retorna un subconjunto de $Q \times \Gamma^*$.
  Formalmente: $\delta \colon Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$
- q_0 denota el estado inicial, donde $q_0 \in Q$
- Z_0 denota el carácter inicial de la pila, donde $Z_0 \in \Gamma$
- F denota un conjunto finito de estados de aceptación.

Máquina de Mealy

Una máquina de Mealy D es una 6-upla $(Q, q_0, \Sigma, \Lambda, T, G)$, donde:

- Q denota un conjunto finito de estados.
- q_0 denota el estado inicial, donde $q_0 \in Q$.
- Σ denota el alfabeto de entrada.
- Λ denota el alfabeto de salida.
- T denota una función de transición, que toma como entrada un estado y un carácter de entrada. Este a su vez retorna un estado de Q.
  Formalmente: $T\colon S \times \Sigma \to S$
- G denota una función de salida que toma como entrada un estado y un carácter de entrada. Este a su vez retorna un carácter de Λ.
  Formalmente: $G\colon S \times \Sigma \to \Lambda$

Diseño de juegos

El diseñador de juegos usualmente tiene a cargo múltiples tareas, solo por listar algunas tenemos:

- Experiencia
- Elementos
- Diseño de nivel
- Control de iteraciones
- Mecánicas
- Curvas de nivel

Diseño de juegos

Schell menciona que entre las tareas fundamentales se encuentran:

- Estética
- Historia
- Mecánicas
- Tecnología

Metodología y desarrollo

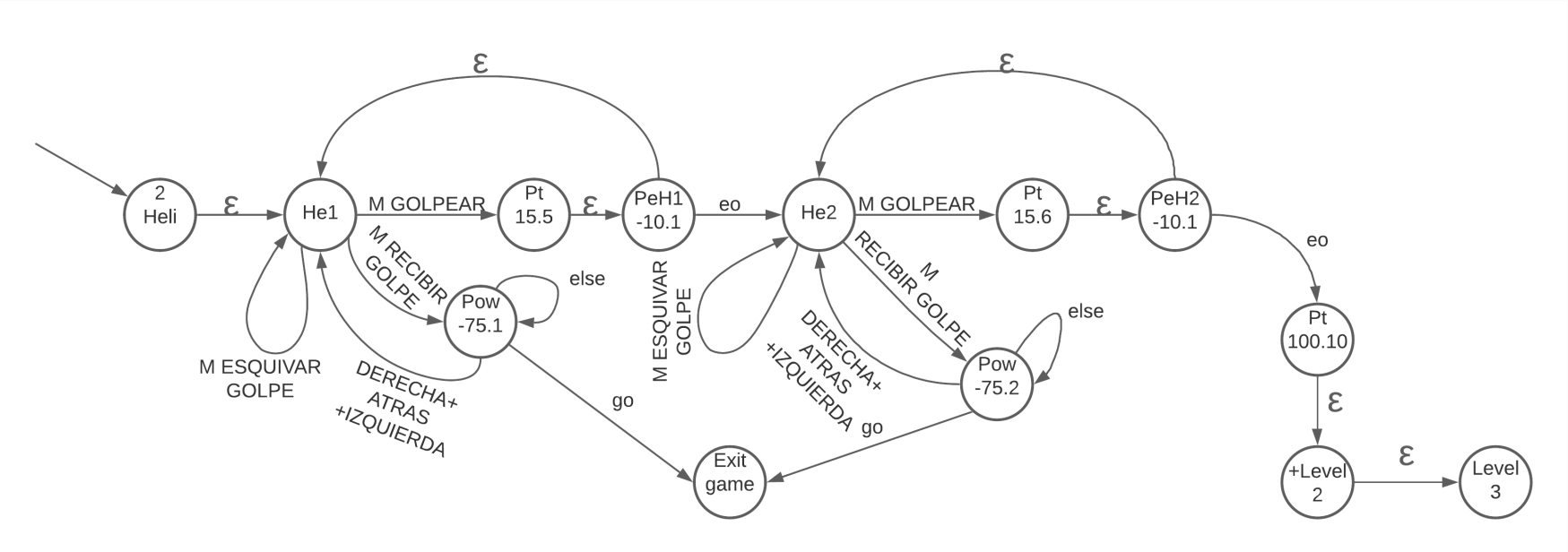
Metodología

Se dividió el proceso en 3 partes

1. Selección de una instancia de juego
2. Diseño usando teoría de autómatas
3. Análisis

Desarrollo

Videojuego Arcade



Diseño de juego de videojuego arcade usando AFND

Videojuego Arcade

Alfabetos de entrada:

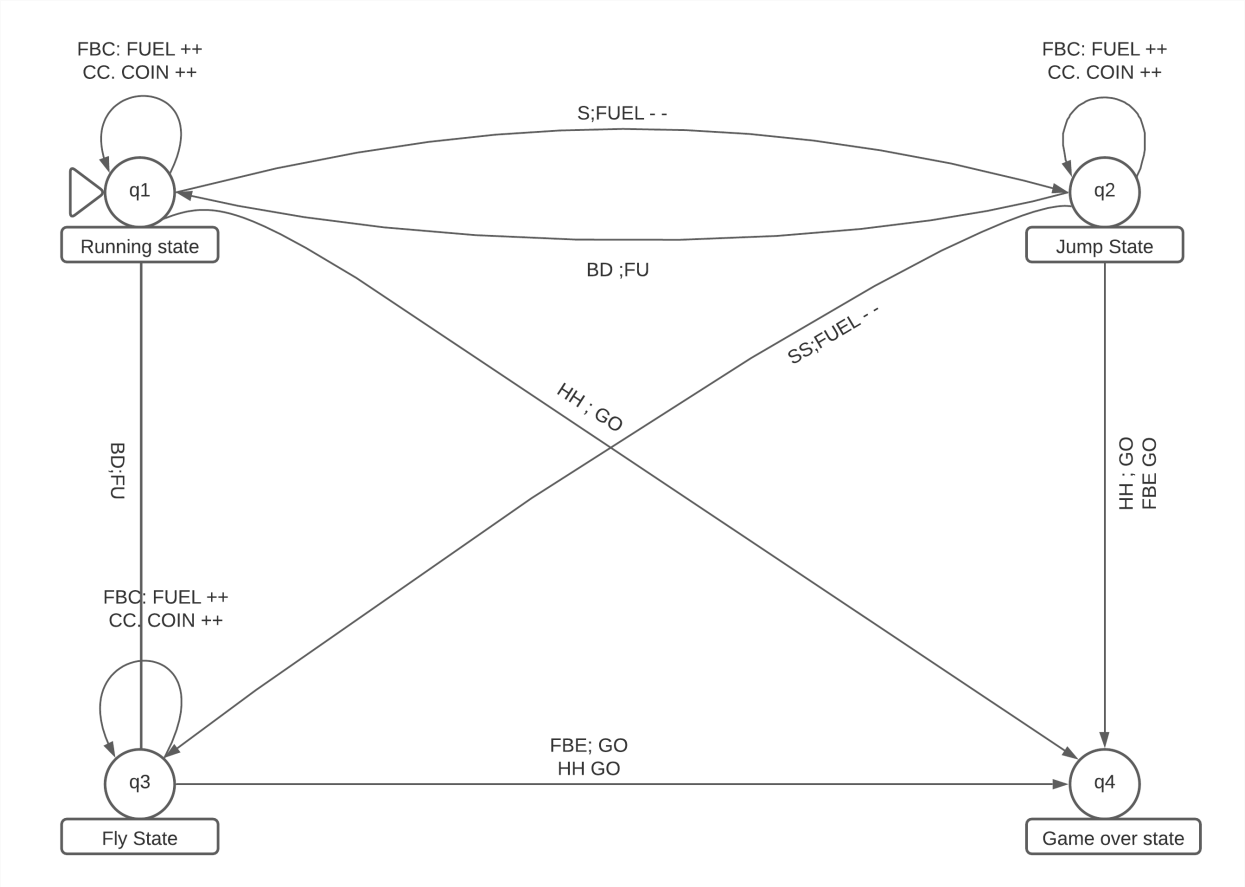
- Σ_{arma} = {K, P, A, H, M, S, C, U, Mi}
- $\Sigma_{seleccion}$ = {g, d}
- $\Sigma_{movimiento}$ = {ADELANTE, DERECHA, ATRAS, IZQUIERDA}
- Σ_{accion} = {GOLPEAR, RECIBIR GOLPE, ESQUIVAR GOLPE}
- Σ_{estado_juego} = {go, gc, gp, eo, Ammo, -Ammo, gz, gd, -gd, gg, -gg} e

Videojuego Arcade

Estados utilizados:

- **Pt**: Se le brindó al jugador algunos puntos.
- **Pow -**: Se redujo los puntos del jugador en una cantidad en especifico.
- **Pow**: Se mantiene la misma cantidad de puntos
- **Pe**: El poder del enemigo es reducido
- **He**: El escenario donde se encuentra el nivel
- **Exit Game**: El juego termina.

Videojuego Infinito de tipo Runner



Diseño de juego de videojuego de tipo Runner usando una Máquina de Mealy

Videojuego Infinito de tipo Runner

Alfabetos de entrada:

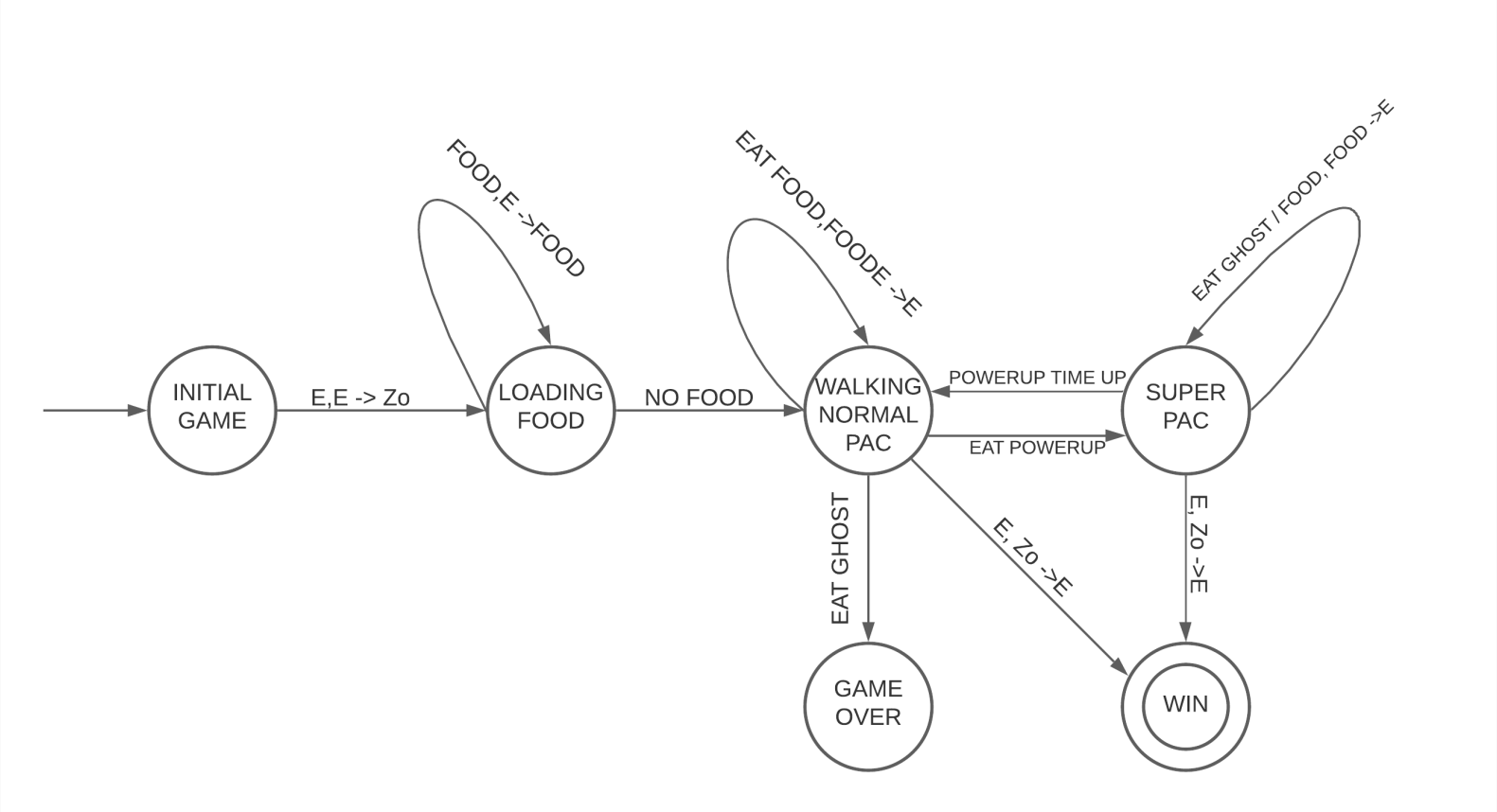
- $\Sigma_{movimiento}$ = {UP_ARROW, DOWN_ARROW, RIGHT_ARROW, LEFT_ARROW}
- Σ_{accion} = {S, SS}
- Σ_{estado_juego} = {GS, GE, CC, HH, FE,-F, C+, D+}

Videojuego Infinito de tipo Runner

Estados utilizados:

- **Running State**: Es el estado inicial que inicializa todos los objetos en la escena de juego.
- **Jump State**: Es el estado donde saltamos y decrece la barra de salud del jugador.
- **Fly State**: Es el estado donde volamos y se reduce el combustible.
- **Game Over State**: Es el estado donde el juego termina, ocurre cuando el jugador golpea un obstáculo o enemigo.

Videojuego Pac-Man



Rediseño del clásico videojuego Pac-Man

Videojuego Pac-Man

Estados utilizados:

- **Initial Game**: Es el estado inicial. En este estado se cargan todos los elementos del juego.
- **Loading Food**: En este estado se ubican los puntos dentro del escenario. Esto varia de nivel a nivel puesto que todos los escenarios no son los mismos.
- **Walking Normal Pac**: Es el estado en el que transcurre la mayor parte del tiempo para el jugador. Consiste en caminar y comer los puntos.
- **Super Pac**: Es el estado en el que se encuentre el jugador después de haber comido el potenciador.
- **Game Over**: Es el estado de fin de juego, ocurre cuando no tenemos un potenciador y chocamos con un enemigo.
- **Win**: Es el estado final del juego, ocurre cuando todos los puntos han sido comidos.

Conclusiones

Conclusiones

- Es posible realizar el diseño de un juego utilizando la teoría formal de autómatas.
- El diseño basado en autómatas puede ayudar el proceso de implementación ya que este estará mas sistemático y organizado.
- El diseño de juegos basado en autómatas es una alternativa a las herramientas clásicas de la ingeniería de software, como los diagramas de caso de uso.
- Si se usa el diseño basado en autómatas, se debe evitar el uso de AFND puesto que esto complica la implementación directa.

Trabajos Futuros

Trabajos Futuros

Se propone las siguientes ideas para trabajos futuros:

- El diseño de juegos basado en autómatas no tiene por que limitarse solamente a videojuegos. Sino también, puede extenderse al diseño de software, ya que como vimos el uso de autómatas puede reemplazar algunas herramientas de la ingeniería de software.
- La implementación y documentación del proceso de desarrollo de un videojuego desde cero, haciendo uso de la metodología presentada en la presente tesis.
- La exploración de otros tipos de autómatas haciendo uso de otras herramientas como inteligencia artificial para el diseño de juegos.
- El estudio y el diseño de mayor cantidad de juegos de distintos géneros para poder hacer una clasificación más precisa de autómata-género de juegos.

Gracias