A Computational Model of Life Cycle of Game Actions and Effect

Frank Appiah

December 18, 2020

A COMPUTATIONAL MODEL OF LIFE CYCLE OF GAME ACTIONS AND EFFECTS.

FRANK APPIAH.

KING' COLLEGE LONDON, SCHOOL OF ENGINEERING, ENGLAND, UNITED KINGDOM.

frank.appiah@kcl.ac.uk

appiahnsiahfrank@gmail.com.

Extended Abstract[+]. Finite State Machine is employed in computationally modeling the life cycle of Maiar game. The focus of this research is computationally modeling the life cycle of a remote command in Maiar game.

Keywords. FSM, machine, computational, game, lifecycle, remote, command, Maiar.

Year of Study: 2016 **Year of Publication: 2020**

1 ***AFFILIATE. UNIVERSITY OF LONDON, KING'S COLLEGE LONDON, DEPARTMENT OF ENGINEERING, LONDON, UK.**

1 INTRODUCTION

The game engine is the heart of the Maiar game software and is developed mainly on interfaces for easy extensibility of many actions and effects. It revolves around actions and effects that show in the 2D graphical user interface with skinning support. Based on a certain query of state of the room and an action inputted by the player/user will in turn then trigger an effect for the player like rendering a different room in the sequence of procedure described in XML data.

Life Cycle of Remote Command

Finite State Machine is used in modeling the life cycle of the remote commands in the game. This is discussed in the next section.

2 GAME MACHINE

A Finite State Machine is good model for computers in describing a regular language in reference to a particular application. It is a mathematical theory of finite automaton with reference to an abstract view. The diagram above is a state diagram concerning a game application running on a computer. The language is a formal definition. It has a set of states and rules for going from one state to another, depending on the input symbol. A finite automaton is a list of five objects

: set of states, input alphabet, rules of moving, start state and accept states. The input alphabet is indicated by input symbols.

A **finite automaton[1]** is a 5-tuple (Q, Σ , δ, q , F), where

1. Q is a finite set called the states,
2. Σ Is a finite set called the alphabet,
3. $\delta: Q \times \Sigma \rightarrow Q$ is the transition function,
4. $q_0 \in Q$ is the start state.
5. $F \subseteq Q$ is the set of accept states.

In describing the Game Machine, the finite automaton GM is formally described as:

1. $Q := \{GUI, CP, PTA, PDE, PPA\}$
2. $\sum := \{PV, TR, VE, EA, PU\}$
3. $\delta : \{transitionfunctions\}$
4. $q_o := \{GUI\}$
5. $F := \{GUI\}$.

GUI : Graphical User Interface
CP : Command Parser
PTA: Player Try-Action
PDE: Player do-Effects
PPA : Player Perform-Actions
PU : Pickup Rock
PV : Present Visual
TR : Take-Rock Action
VE: Visual Effects

EA : Extract Action

The next activity to do is to represent the transition function in a tabulated form. This is called Transition Function Table. The input-ouput states are entered into a table of rows and columns. The delta transition function is characterized as a transition function table shown below:

States	**INPUT SIGNALS**				
	PU	**TR**	**VE**	**EA**	**PV**
GUI	CP	PTA	ε	ε	ε
CP	ε	GUI	ε	ε	ε
PTA	ε	PPA	GUI	PDE	ε

States	INPUT SIGNALS				
	PU	**TR**	**VE**	**EA**	**PV**
PDE	ε	ε	PTA	ε	ε
PPA	ε	ε	ε	PTA	ε

The game controller moves from state to state depending on the input it receives,

1. In the state **GUI**, the controller receiving an input signal **PU** will start to parse commands.
2. In the state **GUI**, on receiving an input signal **TR** will move a state of **PTA**.
3. In the state **GUI**, the controller receiving an input signal **VE** will move to an empty state.
4. With an input signal **EA**, the controller will not move from the **GUI** state to any other state.
5. In the state **GUI**, the controller receiving an input signal **PV** will move to an empty state. **AND**
6. In the state **CP**, the controller receiving an input signal **PU** will move to an empty state.
7. With an input signal **TR** in a state **CP**, the controller will move to a state **GUI**.
8. In the state **CP**, the controller on receiving an input signal, **VE** will not move to any state.
9. In the state **CP,** the game controller on receiving an input signal, **EA** will not move to any state, ε .
10. In the state **CP**, the game controller on receiving an input signal, **PV** will not move to any state, ε . **AND**

11. In the state **PTA**, the controller receiving an input signal **PU** will move to an empty state.
12. With an input signal **TR** in a state **PTA**, the controller will move to a state **PPA**.
13. In the state **PTA**, the controller on receiving an input signal, **VE** will move to a state, **GUI**.
14. In the state **PTA,** the game controller on receiving an input signal, **EA** will move to a state, **PPE**.
15. In the state **PTA**, the game controller on receiving an input signal, **PV** will not move to any state, ε . **AND**
16. In the state **PDE**, the controller receiving an input signal **PU** will move to an empty state.
17. With an input signal **PDE** in a state **CP**, the controller will not move to a state.
18. In the state **PDE**, the controller on receiving an input signal, **VE** will move to a state, **PTA**.
19. In the state **PDE,** the game controller on receiving an input signal, **EA** will not move to any state, ε .
20. In the state **PDE**, the game controller on receiving an input signal, **PV** will not move to any state, ε **AND**
21. In the state **PPA**, the controller receiving an input signal **PU** will move to an empty state.
22. With an input signal **PPA** in a state **CP**, the controller will not move to any state.
23. In the state **PPA**, the controller on receiving an input signal, **VE** will not move to any state, ε .

24. In the state **PPA,** the game controller on receiving an input signal, **EA** will not move to a state, **PTA**.
25. In the state **PPA**, the game controller on receiving an input signal, **PV** will not move to any state, ε .

3 MACHINE PROCESSING

Illustration 1 is a state transition diagram of the Game Automaton(G*A*) called GM_1. It has five states labeled *GUI, CP, PTA , PDE and PPA* and five conditions labeled *PV, VE, PU, TR and EA*. The start state of WFA is GUI and it is normally indicated by a pointing arrow to the state GUI. The accept state is the *GUI* state and it is normally indicated by double circle/round-shape around the state. The arrows moving from one state to another is called transitions. When the automaton receives an input string *{GUI, CP, PTA , PDE, PPA}*, it processes that string and produces an output. The output is either accept or reject. The processing begins in $\mathbf{GM_1}$ start state. The automaton receives the symbols from the input string one by one from left to right. After playing the symbols, GM_1 moves from one state to another along the transition that has symbol as its label. When it plays the last symbol now it is in the accept state. The processing of GM_1 as follows:

1. start in state GUI;
2. play PU, follow transition from GUI to CP;
3. play TR, follow transition from GUI to PTA;
4. play TR, follow transition from CP to GUI;
5. accept because GM_1 is in accept state GUI;
6. play TR, follow transition from PTA to PPA;
7. play VE, follow transition from PTA to GUI;

8. accept because GM_1 is in accept state GUI;
9. play EA, follow transition from PTA to PDE;
10. play VE, follow transition from PDE to PTA;
11. play EA, follow transition from PPA to PTA.

4 TRANSITION FUNCTION

The transition function is used to define the rules of moving. The notation of the transition function is δ(state, input)=state. The transition functions for remote commanding are as follows:

- δ(GUI, PU) = CP
- δ(GUI, TR) = PTA
- δ(CP, TR) = GUI
- δ(GUI, PU) = CP
- δ(PTA, TR) = PPA
- δ(PTA, TR) = CP
- δ(PTA, VE) = GUI
- δ(PTA, EA) = PDE
- δ(PDE, VE) = PTA
- δ(PPA, EA) = PDA.

5 CONCLUSION

This research is a computational model of a game automaton. This game automaton is machine processed to simulate gameplay states. The Finite State Machine, GM is defined formally and a transition function table of state movements in the gameplay is tabulated in 5x5 row-column table. Finally, a rules of movement of the gameplay is cleary enumerated in this work with a notation of transition function.

Compliance with Ethical Standards

(In case of funding) Funding: This is research is funded by King's Alumni Group, Association of Engineering with ISAreference grant number: 204424 20821845.

Conflict of Interest:

Author, Dr. Frank Appiah declares that he has no conflict of interest.

REFERENCES

1. Michael Sipser(1997). Introduction to Theory of Computations. PWS Publishing Company.
2. HopCroft, J. E. and Ulmann, J. D(1979). Introduction to Automata theory, Languages and Computation. Addison Wesley.
3. Roche, E. and Schabes, Y(1997). Finite-State Language Processing. MIT Press.