

Game Rules & State Machines

Prof. Jim Whitehead
CMPS 80K, Winter 2006
January 13, 2006

What are Rules?

- Two trains of thought:
 1. Rules are limitations
 - They limit the allowable actions of a player, thereby creating challenges
 - Example: in the high jump, one cannot use a ladder; in a track race, one cannot run across the midfield; chess pieces have fixed movement (and what you do is move them, not set them on fire, throw them, etc.)
 - But, doesn't cover many rule situations in video games
 - In Legend of Zelda, combat rules are of the form, if the player hits the creature with the sword, the creature dies, or takes damage – no limitation here

What are Rules?

2. Rules set up potential actions.

- That is, rules create affordances
- An *affordance* is a feature that creates possible actions.
 - A handle on a pan affords picking it up while hot
 - A door knob affords twisting and pulling, which allows you to open the door
- Game rules afford certain kinds of action
 - You can view a knight in chess as being limited to just two squares up, and one over
 - Or you can view the knight's rules as affording certain kinds of motion, and the ability to create interlocking support structures with other pieces.
- Rules provide players meaningful actions
- Rules give a game structure
 - By stating what is, and is not, possible
 - Video games need rules that let the characters move as well as rules that prevent reaching the goal immediately

In reality, both

- Games have rules that act as limitations, as well as rules that create affordances
 - In Sonic the Hedgehog:
 - Limitations: many kinds of walls are impassible, Sonic dies if it hits a spike (unless it has some coins), Sonic can only jump so high
 - Affordances: rules specify what happens when Sonic jumps on a spring, goes underwater, collects a coin, and these create possibilities for action

Aspects of Rules

From “Half-real” by Jesper Juul, 2005, pp. 55-56.

- Rules are designed to be above discussion
 - That is, rules should be unambiguous, and able to be implemented without ingenuity
- Rules of a game create a state machine
 - A machine that responds to player action (doesn't require a computer)
- State machine of a game can be visualized as a landscape of possibilities
 - A branching “game tree” of possibilities from moment to moment

Aspects of Rules (cont'd)

- A player must expend effort trying to reach as positive an outcome as possible
 - This creates a challenge
 - The source of many limitations. For example, it is easy to get to the top of a mountain if you take a helicopter, so this isn't allowed.
- The way a game is actually played while the player tries to overcome its challenges is its gameplay.
 - That is, gameplay is the interaction between rules, and players trying to win the game

Aspects of Rules (cont'd)

- Games are learning experiences
 - Players improve their skills at playing the game over multiple playings.
 - At any given time, a player will have a repertoire of skills and methods for overcoming the challenges of a the game
 - A good game continually challenges and makes new demands on the skills of the player.
- Any specific game can be more or less challenging, emphasize specific kinds of challenges, or serve as the pretext for a social event.
 - Different games can create different player experiences

Two Broad Structures of Games

- Games of emergence
 - Interactions among rules combine to create intricate and complex gameplay
 - The historically dominant form of game
 - Chess, Go, Bridge, etc.
 - “Easy to learn but difficult to master”
- Games of progression
 - Challenges are presented serially, by way of special-case rules
 - Designer explicitly determines how the game will progress
 - Examples: Grand Theft Auto, Ratchet and Clank
 - Historically newer game form, evolved out of adventure games

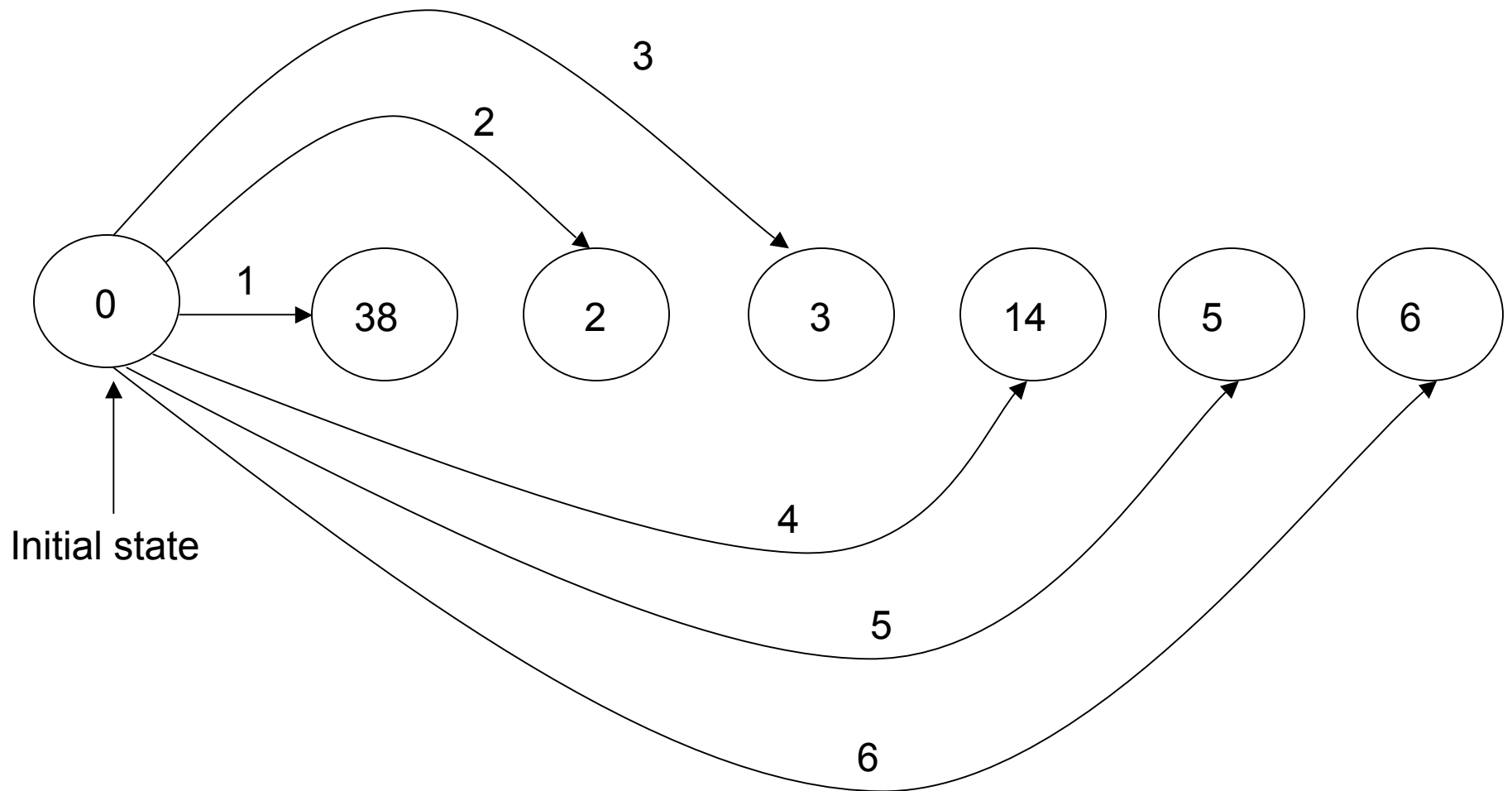
State Machine

- A state machine has:
 - A series of states
 - One of these is the initial state
 - One or more of these are the final (or accepting) state(s)
 - One of these is the current state
 - The current state represents the status of the machine
 - A finite set of possible input events
 - A set of state transition arcs
 - Specifies what happens when you are in a given state, and receive a specific input
 - Also known as the transition function

Example State Machine: Chutes and Ladders

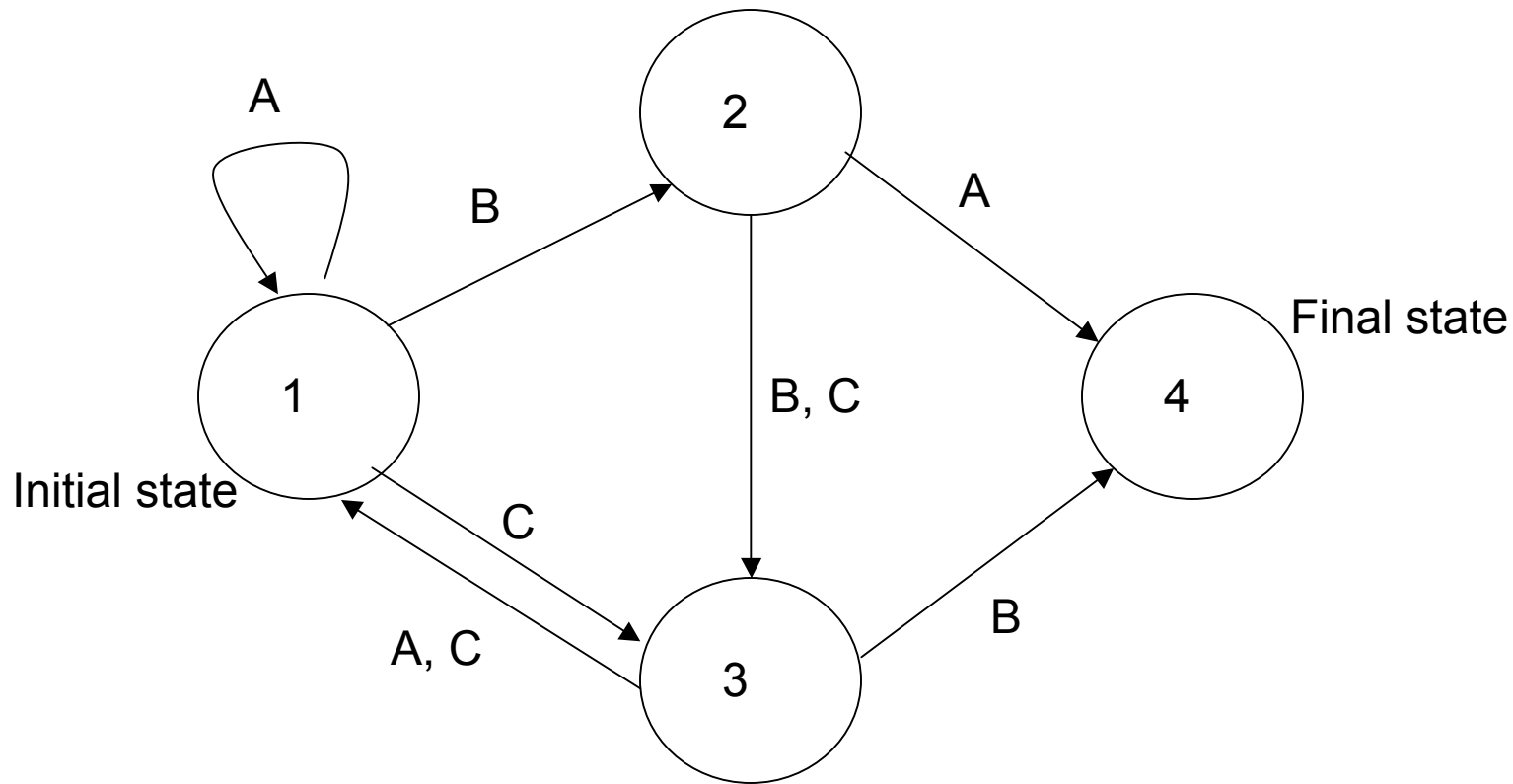
- States: squares on the board, and the space immediately off the board (“space 0”)
- Initial state: the space immediately off the board (space 0)
- Final state: square 100
- Set of input events: #1-6 (from the spinner)
- State transition arcs
 - Usually advancing by the input number of squares, except when there are chutes or ladders

Chutes and Ladders Example



Shows transitions only for state 0

“Living” State Machine



For Inputs: B, C, A, A, C, B, what states do you progress through?

State Machines Are Quite Handy

- When writing software, state machines are a compact way to represent complex interactions between input and current state
 - Can represent the state machine in an array inside the computer
- Can use state machines to perform parsing of computer (and other) languages
- State machines are at the heart of parsers for interactive fiction games
- Entire course on state machines: CS 130

State Machines and Computer Games

- Most computer games don't look like these state machines – what's the connection?
 - Can view the player's current position and attributes (health, possessions) as a very complex state
 - Interactions with the game cause transitions (changes) in this state
 - For example, if the player is at a given position, pressing the right button to move will increase the x value of that position
 - That is, the input of the right button, causes a state transition of increasing the players x position