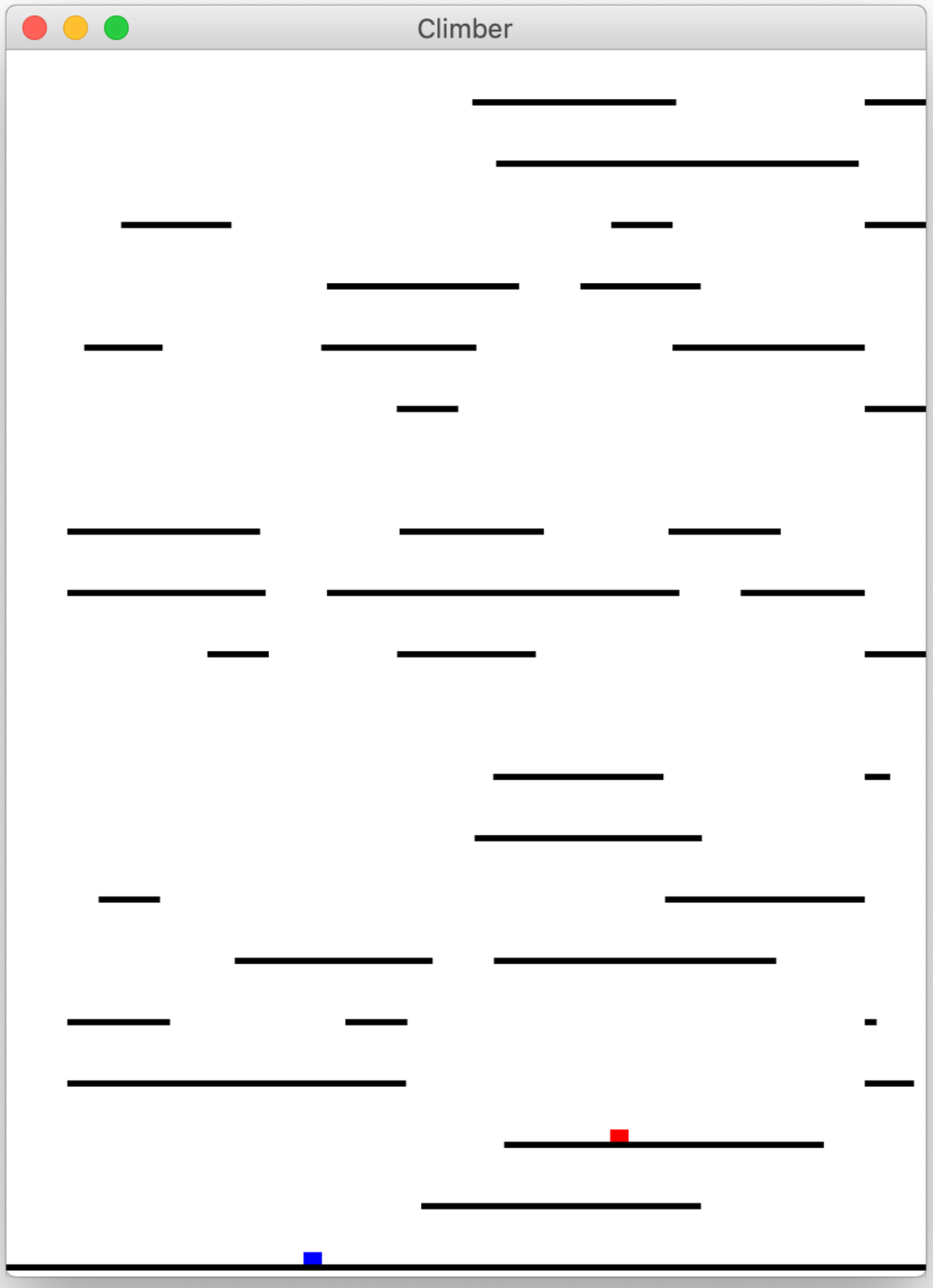
State Pattern

Let's Make a Game

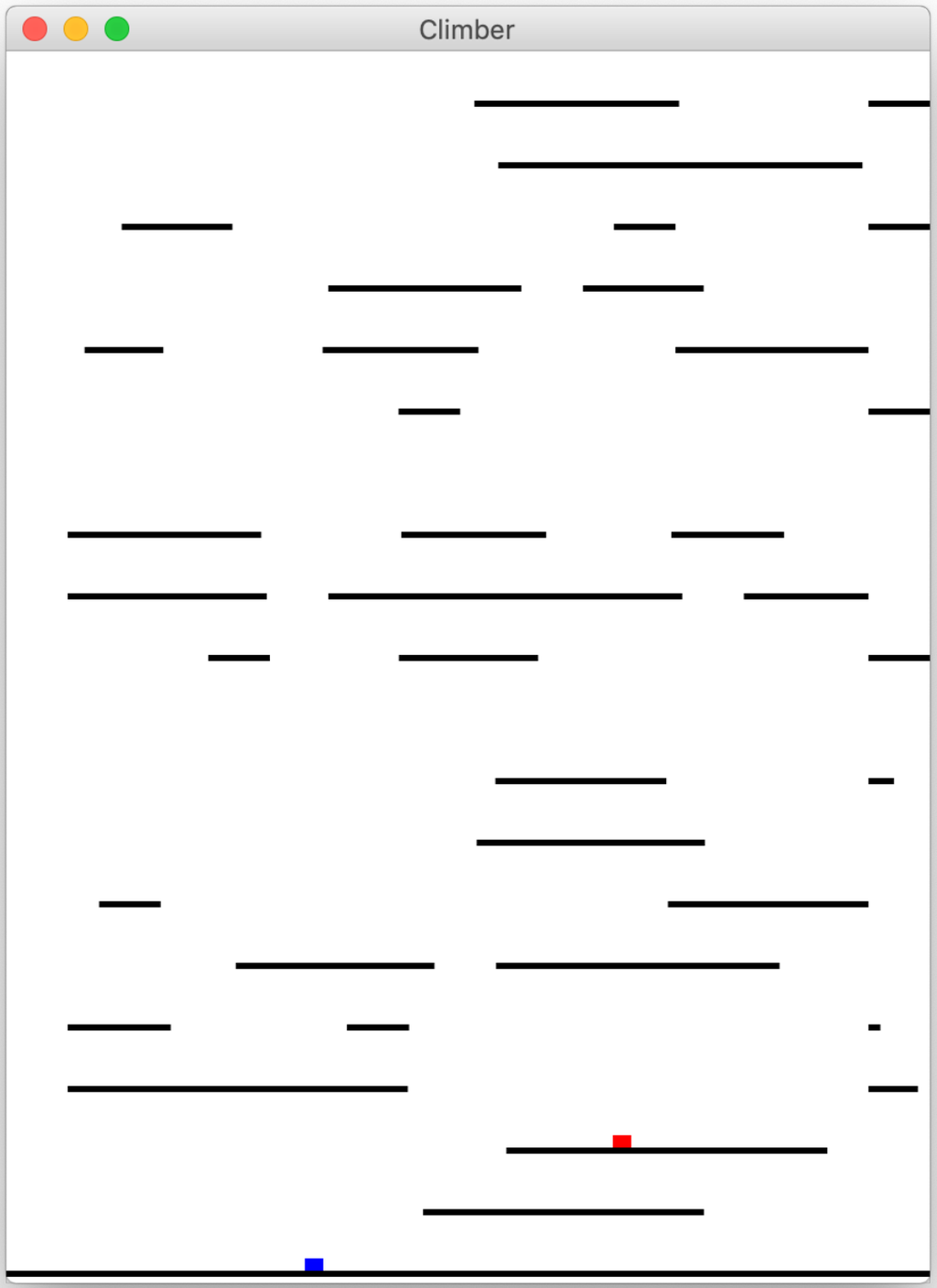
- 2 Player vertical scrolling platform
- Screens scrolls up as the players climb the platforms
- The bottom of the screen is game over
- Goal: Climb faster than the other player



Let's Make a Game

We've already seen

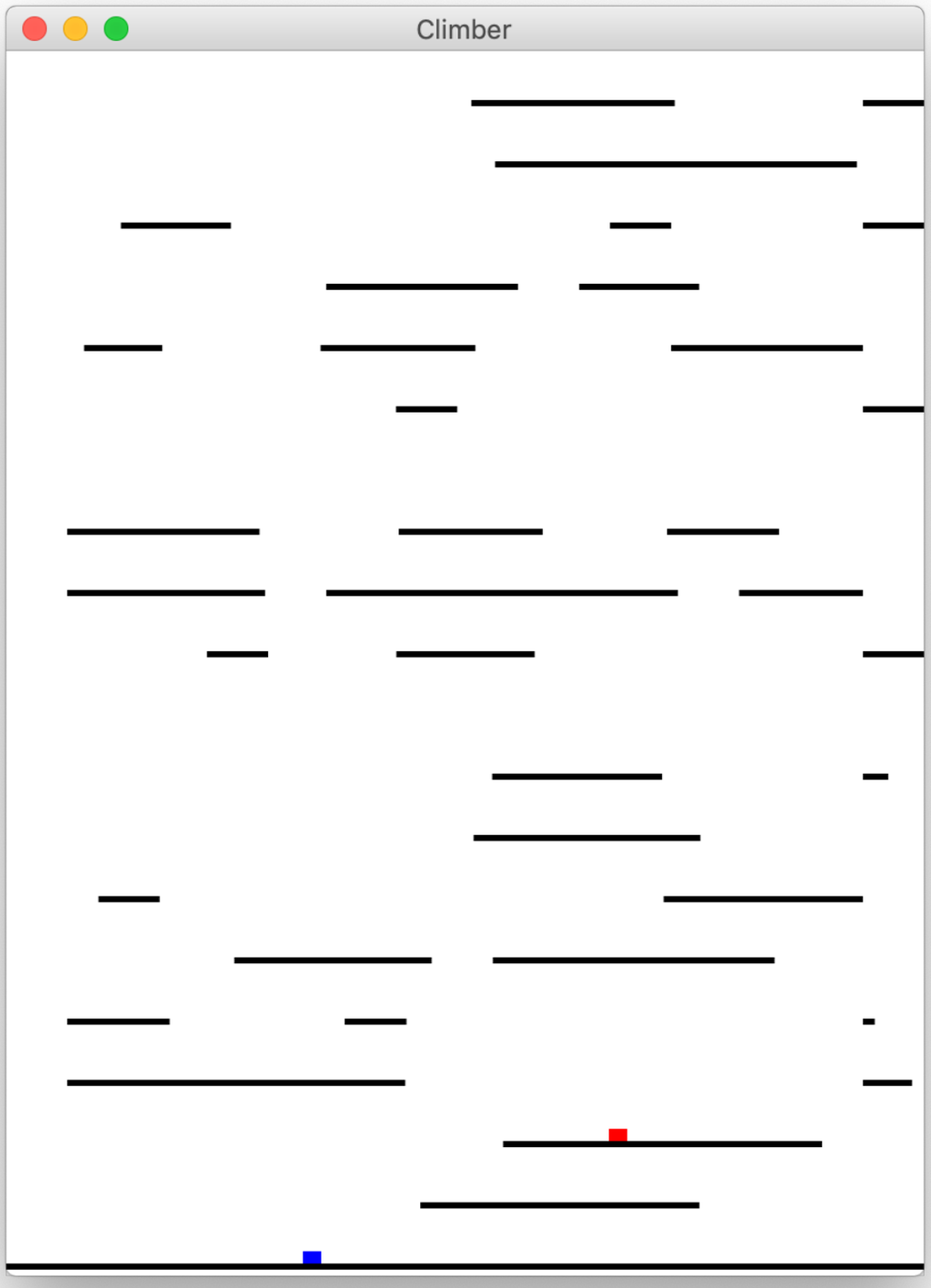
- Physics
  - With modification to allow platforms
- GUI
  - It's all rectangles
  - Update translations as screen scrolls
- Keyboard inputs
- MVC architecture to organize the code



Let's Make a Game

What are we adding today?

- Player behavior
- With states!



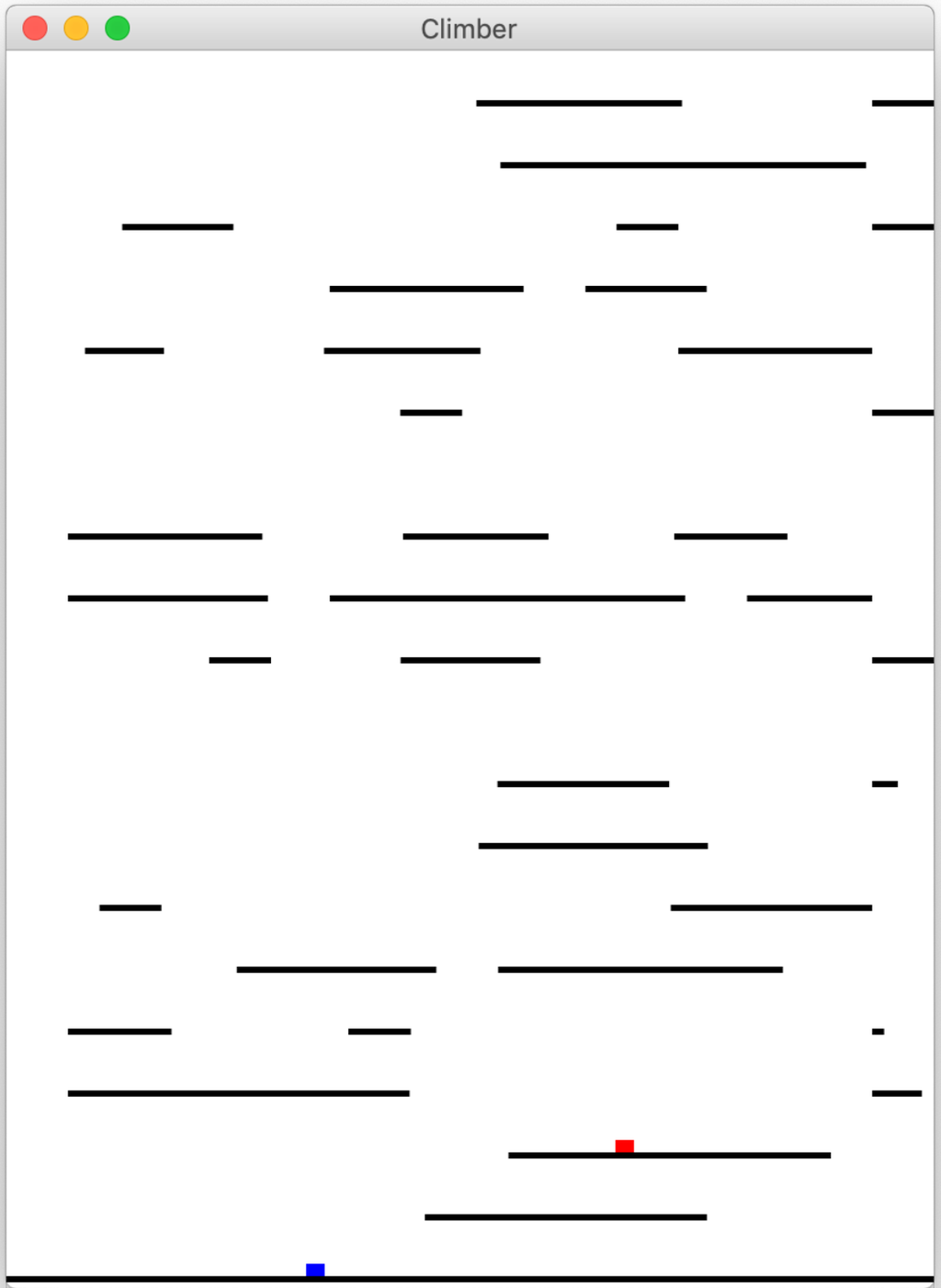
3 Inputs to control each player

- Left
- Right
- Jump

Player behavior

Each player should

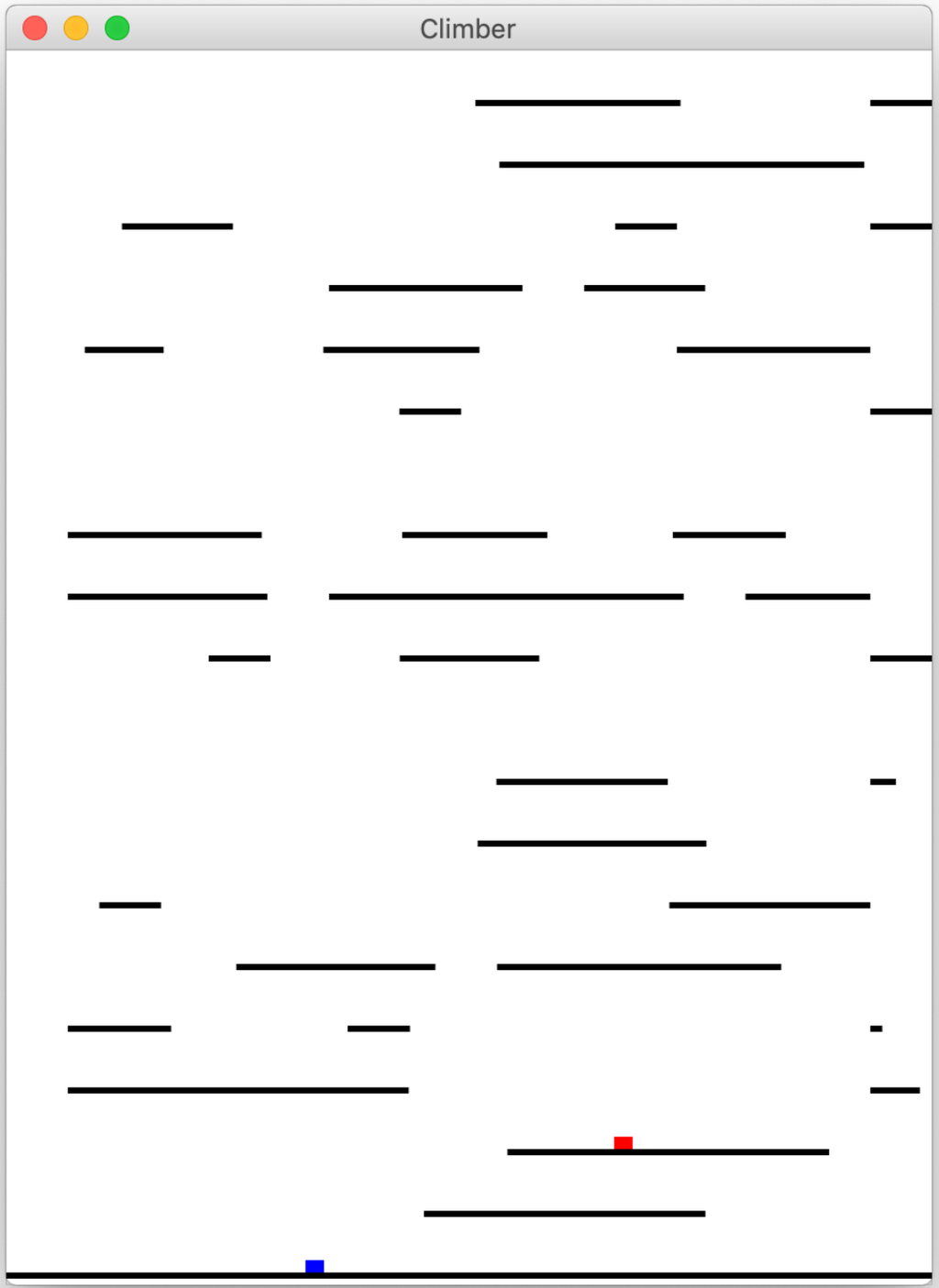
- Walk left and right when keys are pressed
- Jump when jump is pressed
- Jump higher if walking instead of standing still
- Jump at different heights based on how long the jump button is held after a jump
- Move left and right slower while in the air if the direction is changed
- Jump through platforms while jumping up
- Land on platforms while falling down
- Fall if walked off a ledge
- Block all inputs if the bottom of the screen is reached



Player behavior

We could write all this behavior without the state pattern

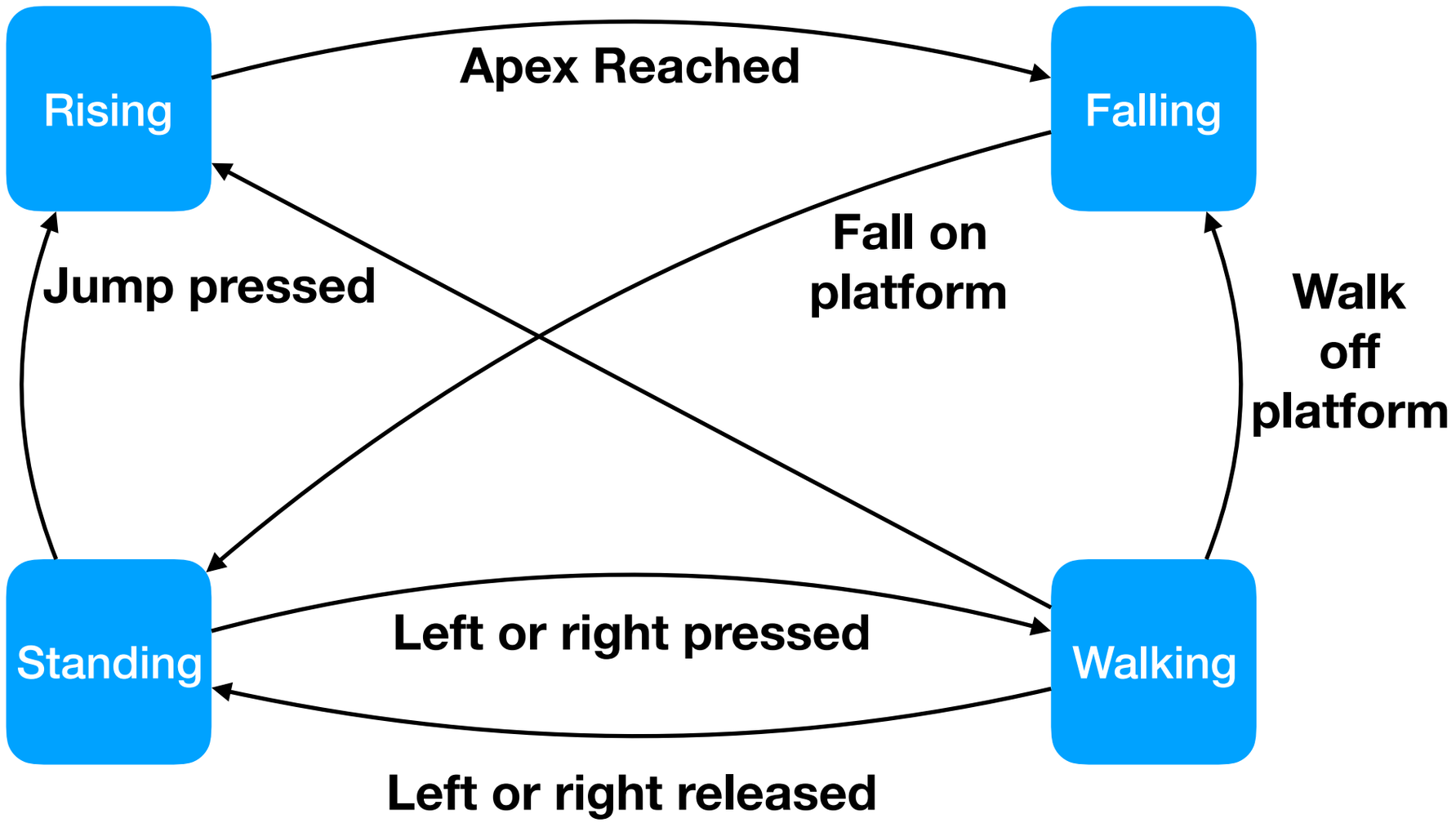
- Code will likely be hard to follow
- Difficult to add new features



Player behavior States

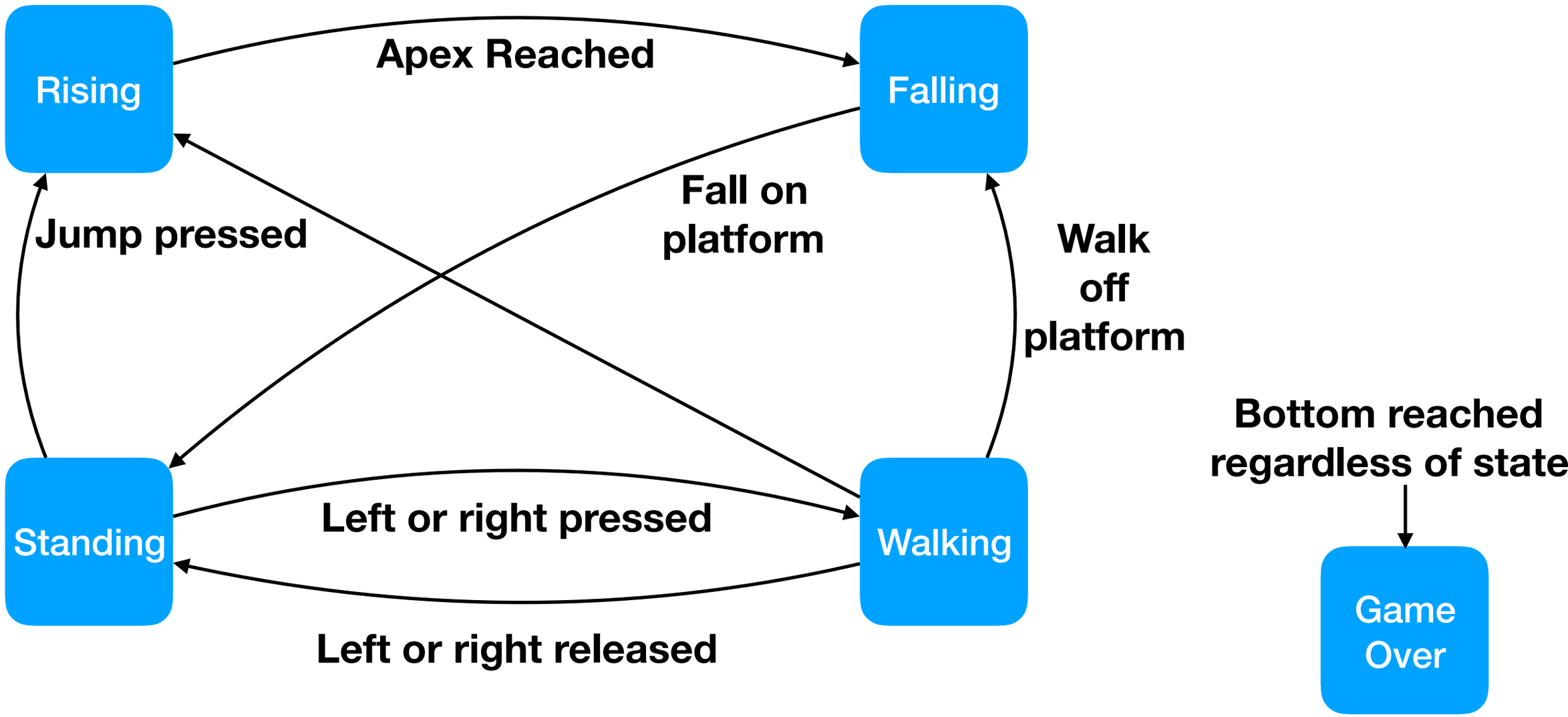
Let's use state

- Think of every possible state of the player that will change behavior
- Draw the transitions between states



Player behavior States

- For each state implement the desired behavior for that state
- Use inheritance to limit duplicate code



Lecture Question

Task: Add a double jump to this game

- Each player can jump 1 additional time while in the air
  - Works whether they are falling or rising
  - Use standing jump velocity for double jumps
  - If a player walks off a ledge they can use their double jump, but do not get 2 jumps
- You have the code in the example repo package "climber"
- Design a way to add this functionality to the game
  - Hint: You'll likely need 2 additional states if you stick with the state pattern
- Implement your design and play the game with your new feature

There is no grader for this question. Submit whatever you have to earn credit

* This question will be open until midnight

Lecture Question

- The intent is to add more states to the state pattern to add this feature
  - By adding state you don't have to worry about breaking the existing code
  - Just add the new states and the state transitions to enter/exit these states
  - Hint [Question spoiler]: New states could be falling after double jump used and rising after double jump used
  - HW Hint: If you use states for this question it should help you when implementing the decimal functionality on the calculator
- Reflection
  - How would you write all the player behavior without states?
  - Do you think the payoff of states is worth it for this game or would you rather use nested conditionals?

* This question will be open until midnight