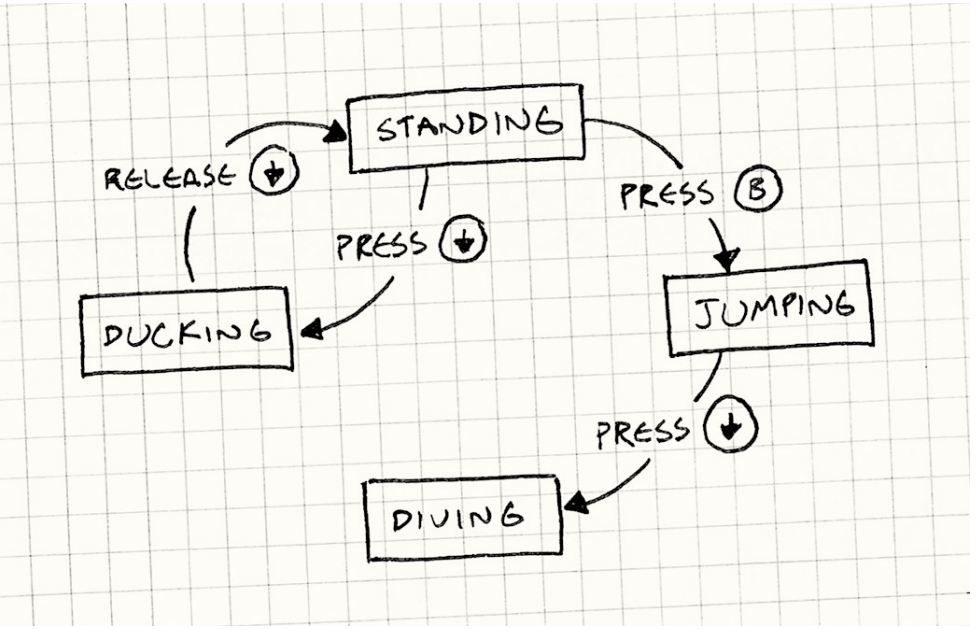
CT3536
(Games Programming using Unity3D)

State Machines
'Psychic Cards' Example

Finite State Machines

- You have a fixed *set of states* of which precisely one is always selected.
  - For our example, a character could be standing, jumping, ducking, diving, or dead;
  - or, a game could be inMenu, inPause, or inGame
- Conditions/events determine the transition between states



- This approach is very appropriate to apply at various times in games: anywhere that your objects need to have different behaviours in different circumstances
- Knowing what state an object is in can be useful in various places, with appropriate code blocks executed (perhaps using a switch statement)
- http://gameprogrammingpatterns.com/state.html

Example: Psychic Cards

This game of “Psychic match pairs” is written entirely using GUI programming in Unity, and takes a State Machine approach.

Each card has a state any any time, with these possible values, which makes programming their behaviour easy and error-free:

- MovingToInitialPosition
- BackFaceUp
- FrontFaceUp
- FlippingToFrontFaceUp
- FlippingToBackFaceUp
- FadingToRemove

The game is controlled via the player’s brain waves: the symbols ”show through” the cards when you focus well

PsychicCard.cs
void Update() *(pseudocode)*

if (state==**CardState.MovingToInitialPosition**)

- Move the card's transform.position a little towards its target position on the table
- When arrived, change state to **CardState.BackFaceUp**

else if (state==**CardState.FlippingToFrontFaceUp**)

- Advance animation (sprite) towards face-up
- When fully face-up:
  - Change state to **CardState.FrontFaceUp**
  - Set symbol on card to maximum opacity
  - If this was the 2nd card (of a pair) to be turned over
    - .. and the pair matches, then change state of both cards to **CardState.FadingToRemove**
    - .. and the pair doesn't match, then change state of both cards to **CardState.FlippingToBackFaceUp**, and reduce 1 player "life"

Continued on next slide…

PsychicCard.cs
void Update() *(pseudocode)*

else if (state==**CardState.FlippingToBackFaceUp**)

- Advance animation (sprite) towards face-down
- When full face-down, change card state to CardState.BackFaceUp

else if (state==**CardState.FadingToRemove**)

- Reduce opacity of card a little
- When at zero opacity, remove from the table and add 1 point to score

else if (state==**CardState.BackFaceUp**)

- Update opacity of card's symbol based on data from the Mindband device

The cards also have an OnClicked() method (which is called when they're clicked as buttons)

- If the card's state is **CardState.BackFaceUp** then change it to **CardState.FlippingToFrontFaceUp**