Lesson 5 - Artificial Intelligence - Teacher's Guide

http://www.stencyl.com/teach/act5/

Overview

Introduce students to the basics of artificial intelligence (AI) within the context of games.

Outcome

Students will take what they've learned and write some simple AI to control non player-controlled characters.

Lesson Plan (1 - 2 hours)

Discussion
15 minutes

Cover the topics under Discussion Notes (Page 2 - End)
Present the topics. Pose questions at appropriate points and encourage students to participate in the discussion.

Activity
45 minutes

Write AI's for a Game
Students will apply what they've just learned to write some AI's for an existing game.

Activity
60 minutes

Work on extra activities
Students will work on a more challenging set of activities in order to demonstrate mastery of the concepts they've learned.

> **Note**: Extra activities are optional but recommended.

Discussion Notes

Topic 1: What is Artificial Intelligence?

In contrast to prior lessons, the teacher will play more a moderator role in guiding a class discussion. There's very little to lecture about here.

> **Discussion Idea**: Before talking, have students come up with their working definition of what artificial intelligence is.
>
> Cite real-life examples, whether from games or not. (Voice Recognition / Siri, Google Search, self-driving vehicles, ...)

In academic terms, Artificial Intelligence (or AI) is the ability for computers to act intelligently. More practically, AI lets a program make **decisions** based on **what it knows** in order to **meet its goals**.

In a board game where the rules are well-defined, the goal could be to get the *highest score* or *defeat the opponent*. In a more open-ended game, the goal could simply be to survive the longest.

For simple games like the ones we're building, we're mostly concerned with creating AI's that fulfill the roles that these actors are playing. They needn't be "intelligent" but merely acting out the way they're expected.

> **Discussion Ideas**: Have students come up with examples of AI from games they've played. Be sure that they dissect the behaviors in specifics, rather than generalities.
>
> After this, have students talk about what makes an AI convincingly real, again, citing both good and bad examples of this.

Topic 2: State Machines

> **Recommendation**: Hold off on this topic until students have reached Step 5 in the activity.

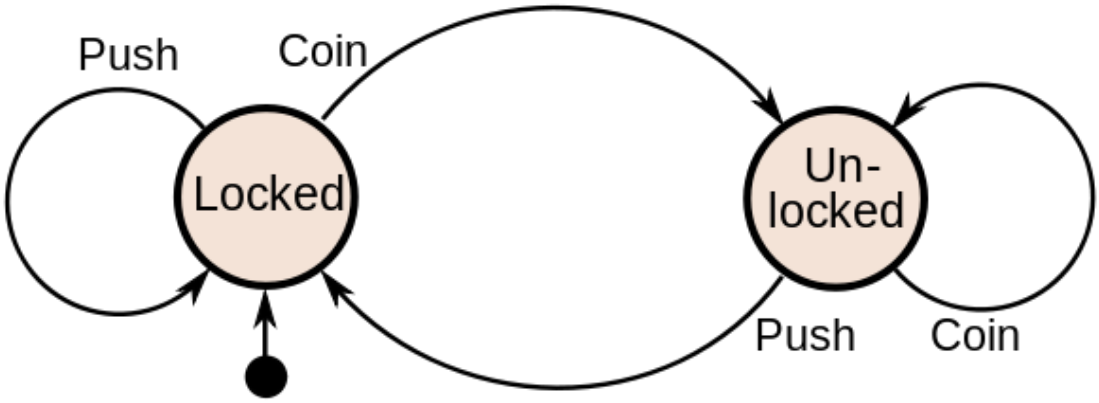
For some actors, these actors follow a simple AI that does the same thing all the time. For example, "seeking" missile will continually follow its target until it reaches it.

For more complex behaviors, this won't work. For example, boss characters usually switch between various modes of attack (or evasion) to keep boss battles challenging and varied.

> **Discussion Idea**: Take a boss from a game of your (or your students') choice and dissect its behavior.
>
> If you don't have one in mind, take this classic example of Bowser from Super Mario Bros.
>
> https://www.youtube.com/watch?v=SeQ2TPu1aVw&feature=youtu.be&t=1m48s

Bowser switches between sliding around on the ground, spitting fire, jumping and throwing hammers.

To do this, we use what's called a State Machine.



State Machines consist of various states and connections between these states. Within a state, an actor will act a certain way (and sometimes look a certain way). Switching between states happens when certain conditions that you specify are met.

TODO: Replace this with something game-oriented in the future.

In the example above (which represents a turnstile in an amusement park or subway), the state machine consists of 2 states.

- Locked
- Unlocked

The turnstile starts in the **Locked** state. Two actions are available for exiting this state - you can...

- **Push** the turnstile
- Insert a **Coin**

Pushing, predictably gets you nowhere because you need to pay. Inserting a coin will unlock the turnstile, thereby transitioning it to the **Unlocked** state.

Tip: If necessary, for completeness, repeat this analysis on the Unlocked state.

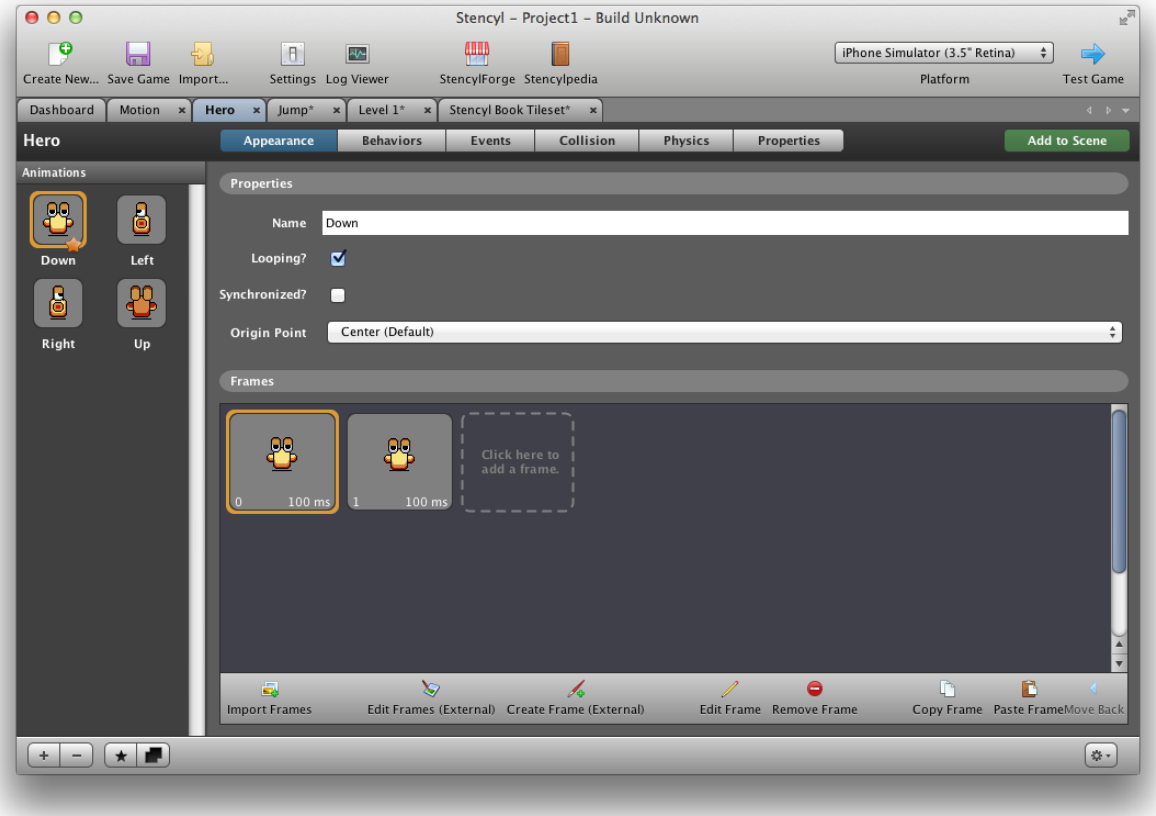
Topic 3: State Machines = Animations in Stencyl

How does this apply to Stencyl and game making?

In Stencyl, state machines exist as “Animation (States)” for Actors.

> **Aside**: In MIT Scratch, they are called Costumes.

Animations are specified on an Actor’s Appearance page. Each Animation corresponds to a state.



In the example above, the Hero has 4 states - Up, Down, Left and Right that correspond with the direction he’s facing.

Switching animations is done using the following block under Actor > Draw.

The current animation is retrieved using the following block, on the same page.

Putting this all together, one could do the following to have the code react differently

depending on what state the Hero was in.