A finite number of states

The machine is in a single state at a time

Each state has a set of transitions to other states

Each transition is associated with an input
A finite number of states

The machine is in a single state at a time

Each state has a set of transitions to other states

Each transition is associated with an input
A finite number of states

The machine is in a single state at a time

Each state has a set of transitions to other states

Each transition is associated with an input
A finite number of states

The machine is in a single state at a time

Each state has a set of transitions to other states

Each transition is associated with an input
A finite number of states

The machine is in a single state at a time

Each state has a set of transitions to other states

Each transition is associated with an input
FINITE STATE MACHINE
STATE PATTERN

Allow an object to alter its behaviour when its internal state changes. The object will appear to change its class.
STATE

Useful for:

- Character controllers
- Menu systems
- Interactable objects
- Animations
- Most frequently executed code paths with a long switch statement or a lot of if/else sentences.
## State

**Useful for:**

- Character controllers
- Menu systems
- Interactable objects
- Animations
- Most frequently executed code paths with a long switch statement or a lot of if/else sentences.
STATE

Useful for:

- Character controllers
- **Menu systems**
- Interactable objects
- Animations
- Most frequently executed code paths with a long switch statement or a lot of if/else sentences.
Useful for:

- Character controllers
- Menu systems
- **Interactable objects**
- Animations
- Most frequently executed code paths with a long switch statement or a lot of if/else sentences.
Useful for:

- Character controllers
- Menu systems
- Interactable objects
- Animations
- Most frequently executed code paths with a long switch statement or a lot of if/else sentences.
STATE

Useful for:

- Character controllers
- Menu systems
- Interactable objects
- Animations

Most frequently executed code paths with a long switch statement or a lot of if/else sentences.
COMMON STATE METHODS

Enter
Call when transitioning into the state
Example uses: save timestamp, set graphics, play audio

Exit
Call when transitioning out of the state
Example uses: clean up, set graphics, play audio
COMMON STATE METHODS

**Enter**

Call when transitioning into the state

Example uses: save timestamp, set graphics, play audio

**Exit**

Call when transitioning out of the state

Example uses: clean up, set graphics, play audio
COMMON STATE METHODS

Enter

Call when transitioning into the state

Example uses: save timestamp, set graphics, play audio

Exit

Call when transitioning out of the state

Example uses: clean up, set graphics, play audio
COMMON STATE METHODS

Input actions

- Player input
- Update pattern
- Interaction with other object
Concurrent State Machines

I.e. use of multiple state machines

Example –
- What a game character is doing
- What a game character is carrying
Concurrent State Machines

I.e use of multiple state machines

Example –
- What a game character is doing
- What a game character is carrying
Concurrent State Machines

I.e. use of multiple state machines

Example –
- What a game character is doing
- What a game character is carrying
Hierarchical State Machines

States are organized in a hierarchy

Can pass handling of an input to parent
Hierarchical State Machines

- States are organized in a hierarchy
- Can pass handling of an input to parent
Hierarchical State Machines

States are organized in a hierarchy

Can pass handling of an input to parent
Pushdown Automata

A stack of states
Pushdown Automata

A stack of states
Task: Refactoring replay

Use the state and class diagrams in the task description to refactor the macro/replay system.

You need to finish the flyweight replay, before you can start on the refactoring task.

The deadline is one week longer than on other tasks.

Read Game Programming Patterns state chapter and materials about State in CGLearn.
HOMEWORK

Task: Refactoring replay

Use the state and class diagrams in the task description to refactor the macro/replay system.

You need to finish the flyweight replay, before you can start on the refactoring task.

The deadline is one week longer than on other tasks.

Read Game Programming Patterns state chapter and materials about State in CGLearn.
HOMEWORK

Task: Refactoring replay

Use the state and class diagrams in the task description to refactor the macro/replay system.

You need to finish the flyweight replay, before you can start on the refactoring task.

The deadline is one week longer than on other tasks.

Read Game Programming Patterns state chapter and materials about State in CGLearn.
HOMEWORK

Task: Refactoring replay

Use the state and class diagrams in the task description to refactor the macro/replay system.

You need to finish the flyweight replay, before you can start on the refactoring task.

The deadline is one week longer than on other tasks.

Read Game Programming Patterns state chapter and materials about State in CGLearn.
Task: Refactoring replay

Use the state and class diagrams in the task description to refactor the macro/replay system.

You need to finish the flyweight replay, before you can start on the refactoring task.

The deadline is one week longer than on other tasks.

Read Game Programming Patterns state chapter and materials about State in CGLearn.
HOMEWORK

Task: Refactoring replay

Use the state and class diagrams in the task description to refactor the macro/replay system.

You need to finish the flyweight replay, before you can start on the refactoring task.

The deadline is one week longer than on other tasks.

Read Game Programming Patterns state chapter and materials about State in CGLearn.