Workbook

Version 1.0

# Version 1.0

## Features

* Ability to move in all directions.
* Ability to sprint.
* Ability to jump.
* Ability to crouch.
* Cannot sprint while crouching and vice versa.
* Can only sprint while moving forward.
* Jumping will cancel sprinting or crouching.
* Cannot sprint or crouch during a jump.
* Player will need to hold down the sprint button to sprint.
* Player will only need to press the crouch button to toggle it on or off.
* A limited amount of energy to be used for sprinting and other activities.
* Energy will be regenerated fastest when idle. ¾ as fast when moving. Not at all when jumping or sprinting.
* Sprinting will increase movement speed.
* Crouching will decrease movement speed.

## Requirements

Use Case Diagram:



|  |  |
| --- | --- |
| Use Case Name | Move |
| Participating Actors | Initiated by Player |
| Flow of Events | 1. The Player activates the “Move” function during a game session. 2. The system sets the character state to “Moving” and changes the attributes accordingly. |
| Entry Condition | * Player currently in a game session. |
| Exit Conditions | * The use case successfully completes a flow of events. |
| Quality Requirements | * System must respond within 0.02 seconds. |

|  |  |
| --- | --- |
| Use Case Name | Sprint |
| Participating Actors | Initiated by Player |
| Flow of Events | 1. The Player activates the “Sprint” function during a game session. 2. The system checks to see if the “Sprint” function is available at that time. It does this by checking if the character is not jumping and has sufficient energy and is moving forward.    1. If it is available, change the character’s state to “Sprinting” and change attributes accordingly.    2. If it is not available, do nothing and exit use case. |
| Entry Condition | * Player currently in a game session. |
| Exit Conditions | * The use case successfully completes a flow of events. |
| Quality Requirements | * System must respond within 0.02 seconds. |

|  |  |
| --- | --- |
| Use Case Name | Jump |
| Participating Actors | Initiated by Player |
| Flow of Events | 1. The player activates the “Jump” function during a game session. 2. The system checks to see if the “Jump” function is available at that time. It does this by checking if the character is already not jumping.    1. If it is available, change the character’s state to “Jumping” and change attributes accordingly.    2. If it is not available, do nothing and exit use case. |
| Entry Condition | * Player currently in a game session. |
| Exit Conditions | * The use case successfully completes a flow of events. |
| Quality Requirements | * System must respond within 0.02 seconds. |

|  |  |
| --- | --- |
| Use Case Name | Crouch |
| Participating Actors | Initiated by Player |
| Flow of Events | 1. The player activates the “Crouch” function during a game session. 2. The system checks to see if the “Crouch” function is available at that time. It does this by checking if the character is not jumping.    1. If it is available, the system checks to see if the character is already crouching.       1. If the character’s state is already “Crouching”, change state to “Standing” and change attributes accordingly.       2. If the character’s state is “Standing”, change state to “Crouching” and change attributes accordingly.    2. If it is not available, do nothing and exit use case. |
| Entry Condition | * Player currently in a game session. |
| Exit Conditions | * The use case successfully completes a flow of events. |
| Quality Requirements | * System must respond within 0.02 seconds. |

## Analysis and Design

Looking at the Move use case, we can create the following state chart:



Looking at the Sprint use case, we can derive the following state chart:



Where complete could be when energy runs out or use lets go of the sprint button.

Looking at the Jump use case, we can derive the following state chart:



Looking at the Crouch use case, we can derive the following state chart:



To put it all together, let us look at the possible combined states:

* Standing can go with Idle, Moving and Sprinting.
* Crouching can go with Idle and Moving.
* Jumping can go with Idle and Moving. (Idle Jump is up and down, where moving jump has a difference in x and y directions.)

This leaves us with seven states:

* Standing, Idle
* Standing, Moving
* Standing, Sprinting
* Jumping, Moving
* Jumping, Idle
* Crouching, Moving
* Crouching, Idle.

The resulting state chart for Character Movement:



To capture this state chart in the design, let’s use a State Design Pattern. The resulting class diagram is:



Looking at the above use cases, we can derive the following classes:

* State
* Character
* Attributes

Putting it all together, this is the structure of the Character State Script:

