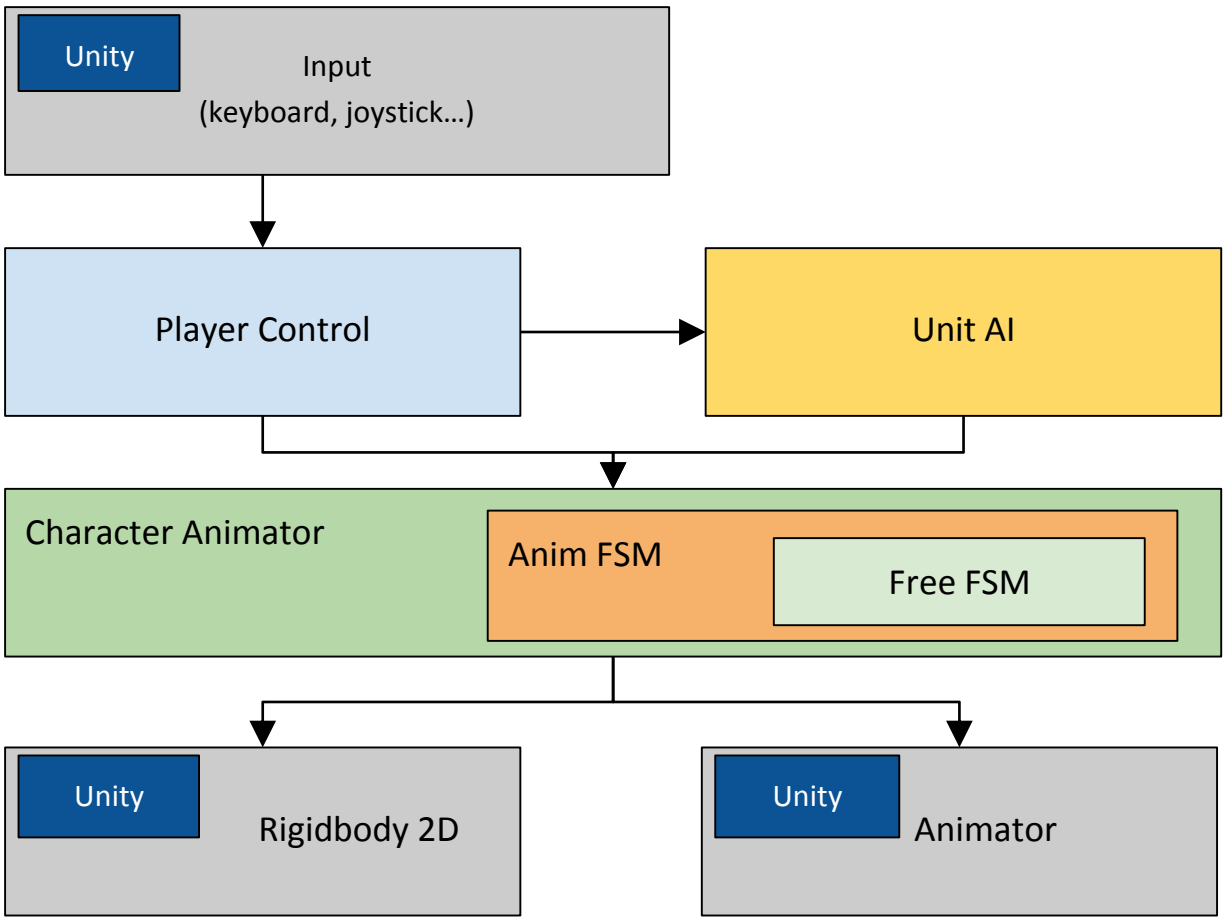
Introduction.

Our goal was to create simple movement control component and approved animations with it. Since animations are managed through Unity Animator you can use any Unity friendly animation.

Architecture

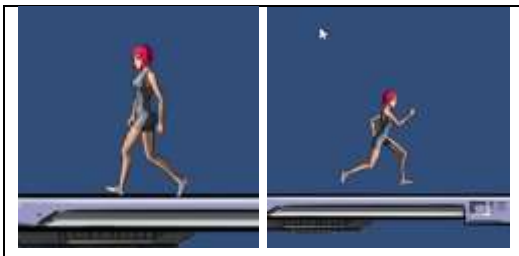
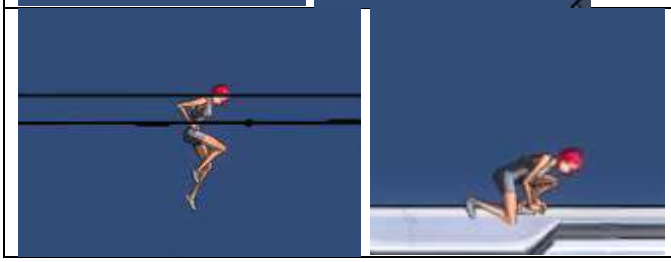


Player Control interprets control signals and events from player input device into the Character Animator commands to control movements. Also Player Control can recognize complicated control commands and pass character's AI control.

Unit AI implements complex automatic character control programs. It is supposed to use Unit AI to control AI NPC. At the moment it implements moving with platform edge detecting during the time which is setted by Player Control.

Character Animator realizes character's location control depending on the control signals. Also establishes character's animation according to the current movement. This is realized by included object Anim FSM. Anim FSM sets Unity Animator state by determining movement parameters.

What we have

	Walking and running. Character can move horizontally depending on player control while standing on the ground.
	Jumping up, without side velocity.
	Jumping to the side and fall with side velocity.
	Clinging to the edge of the platform and pulling up on the platform, both on the fly and from the ground.
	Landing after fall.

Player Control Overview

With Unity Input we define a joystick deviation value and if this value above a certain threshold we decide to move the character left or right. Movement begins with the Unity AI command which is walking to the correct direction. If Player also holds joystick more than a certain time then stop the AI and directly from Player Control make the character run. If Player resets joystick deviation than character goes a short distance under AI control and stops.

Also track “Jump” button to jumping up or to side jumping if button was pressed during the movement; track “Walk” button to switch from “running” to “walking” and other commands.

Unit AI Overview

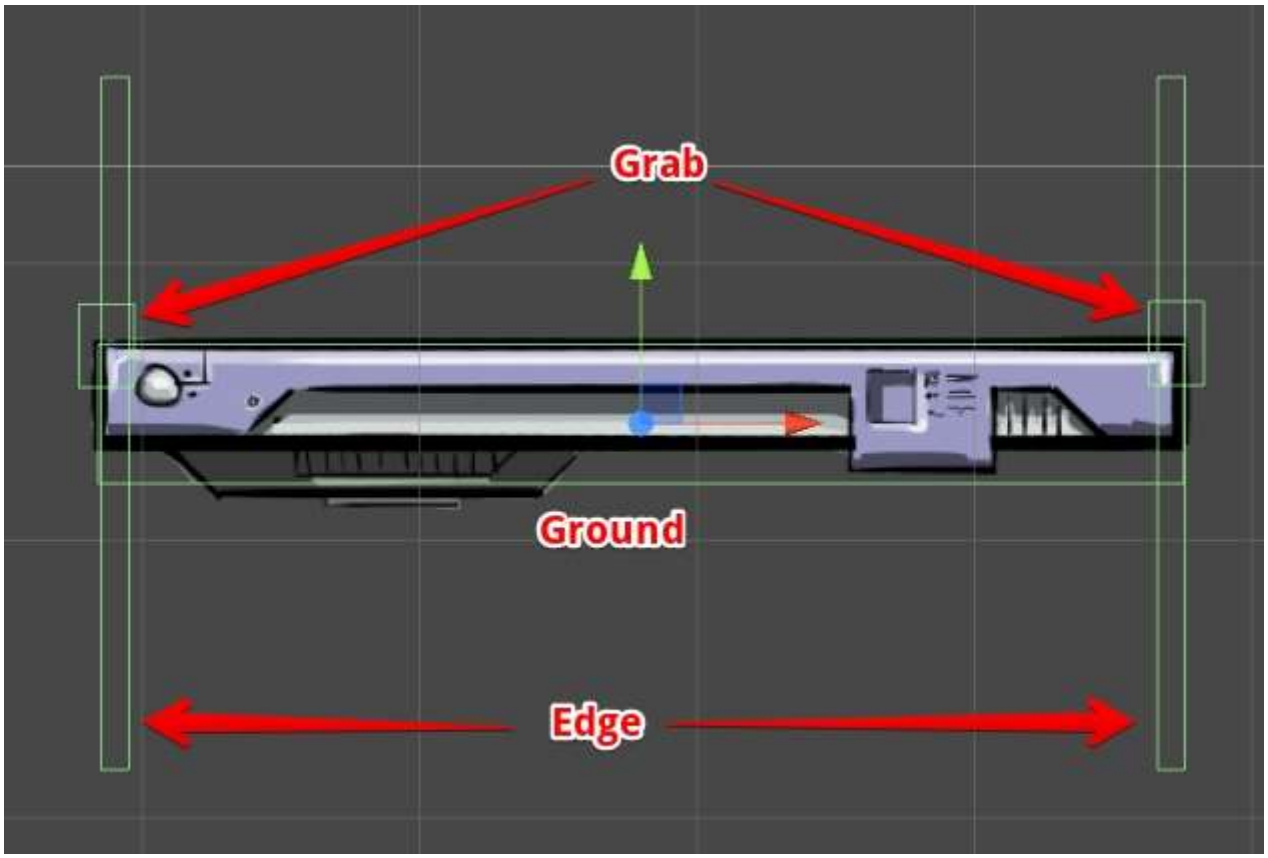
The purpose of this script is a complex character control with or without the player's control.

Now it is a finite state machine and two states: inactivity (CAIState_Idle) and movement during the setted time (CAIState_Walk). In the second state AI moves on command from the Player Control. The command indicate where, how long, walking or running.

AI defines the edge of the platform using a set of specific subordinate objects GameObject (List<Transform> StopCheckers). Their intersection with special platform’s colliders (Edge layer) gives us information that the character on the edge.

AI stops a character if predetermined time of standing on the edge of platform has passed. If a movement start has already begun on the edge colliders edge detection is not carried out for some time.

Platform colliders diagram.



Colliders are named by the names of their layers.

Collider “Ground” is used to determine the surface player can walk on and which can stretch.

Collider “Edge” is used to determine the edge of the platform, not only when the character is on the platform but underneath for easy positioning at the point of clinging.

“Grab” colliders are used to determine that the character is in position to cling a platform and climb on it.

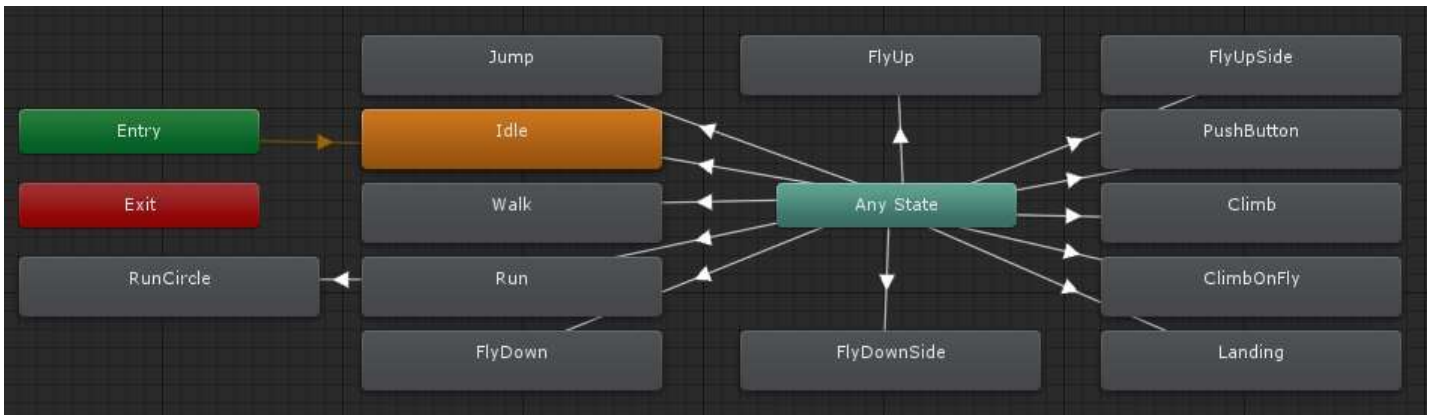
Character Animator Overview

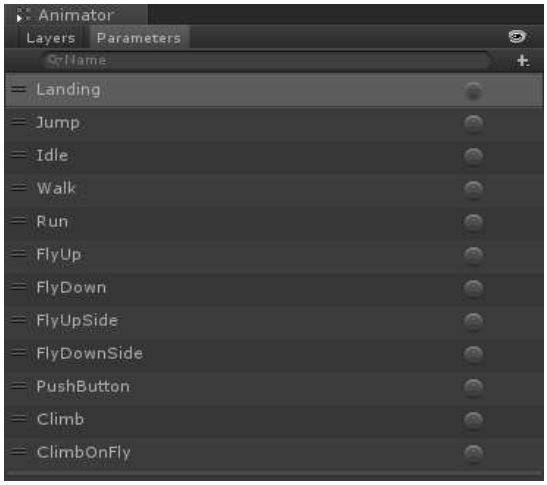
It performs all the work on the movement and character animation. Object state changing is carried out in the method void FixedUpdate () inherited from MonoBehaviour.

In addition to determine whether we are on ground and update animation machine script also performs one of three updates: climb on the platform after clinging update, fly update and movement on the ground update.

Animation Control

Control is a correct animation launching in accordance with the current state of the character considering switching between animations features. For this Character Animator contains Anim FSM that represents a state machine. Each state in state machine matches Animator Unity state configured to a character. Switching to a new state causes an Animator’s trigger with the same name, after that Animator switches its states and character’s animations.

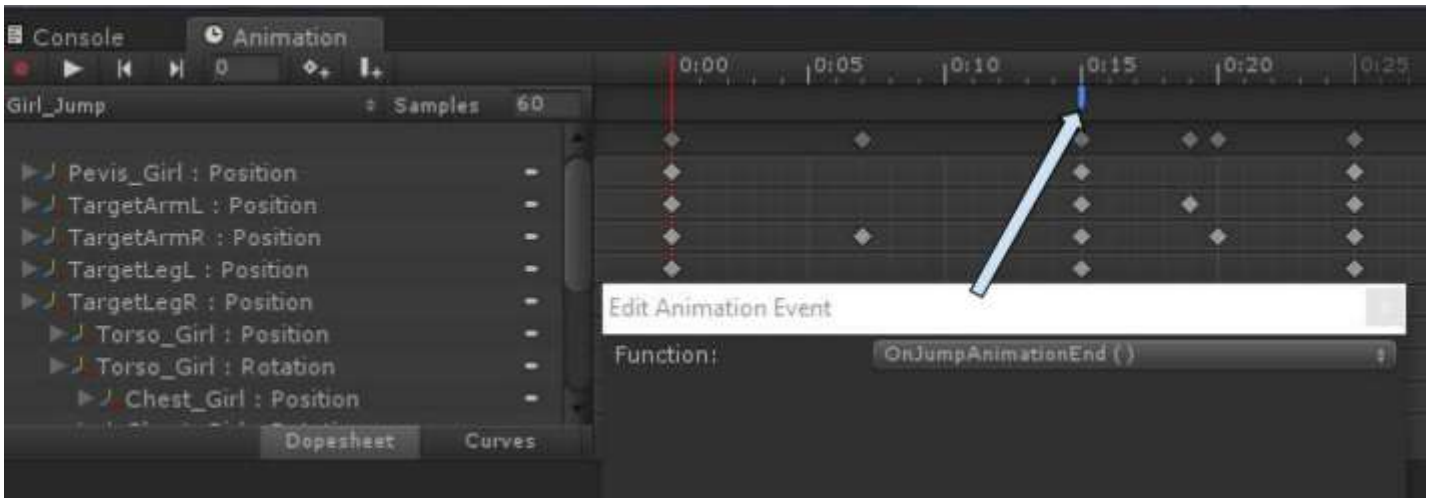




AnimFSM on its update determines using different parameters which state is the most appropriate and tries to switch to this state. You can not switch if:

- AnimFSM is in the same state
- Time isn’t over since the previous switch
- Switching to this state is prohibited from switch settings
- The current state is locked

State is locked when animations that can not be interrupted are playing. For example: climbing on the platform. Special event which is called in the end of each animation remove locking from the state.



Example: Event Character Animator is called in the end of Girl_Jump animation.

Movement on the ground update

If the script does not perform the animation during which you can not move (eg landing after a fall from a great height), it tries to move the character in the predetermined direction.

If the character is up against a wall, then 1. the script sets the state of movement (_move) in the state of standing, 2. it estimates how much time player makes a character rest against the wall and if this time is more than the required threshold, script tries to launch climbing on the obstacle (TryClimbFromGround()).

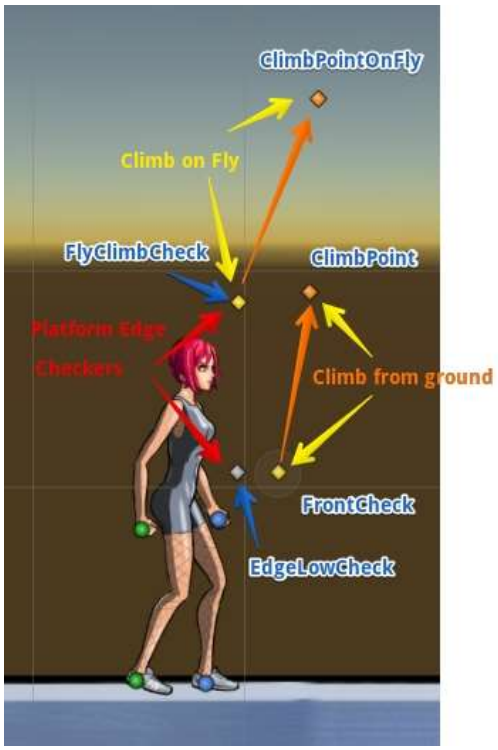
If nothing prevents movement script sets necessary velocity to the component of RigidBody2D object.

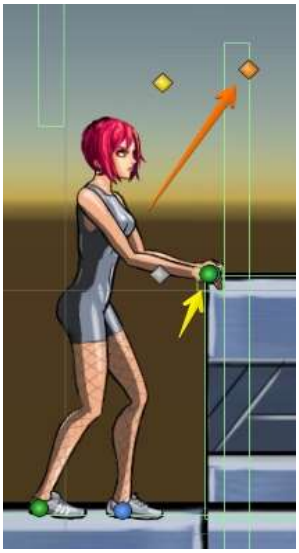
Synchronization of movement and leg movements performed by changing the speed of playback animation in the Unity Animator, for example in state Walk or Run.

Fly Update

Vertically moving velocity is setted less than the maximum velocity. If the automatic clinging to the edge flag is determined script tries to find edges and cling. If not script tries to cling after Player indicated to do so horizontally moving joystick.

Climb on the platform update





There are two ways to switch to this state: from the ground (bool TryClimbFromGround()) and on the fly(bool TryClimbOnFly()). To do this character object has dependent GameObjects: FrontCheck and FlyClimbCheck. When one of them is in contact with grab collider of a platform character may start climbing.

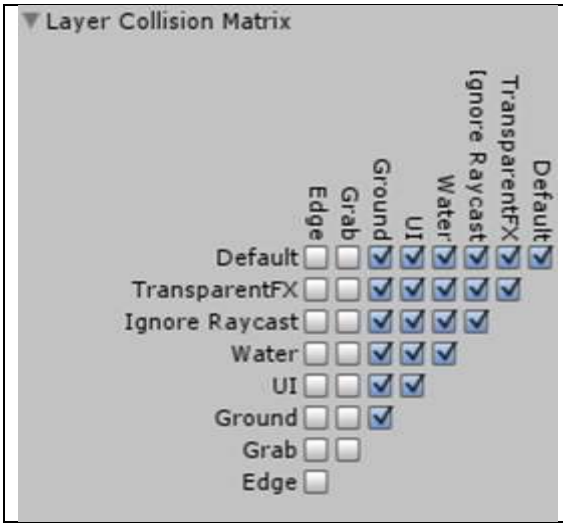
Climbing is even in time process of moving from the climbing start position to the point above the platform. In the end character stands at this point.

Climbing has 2 parts:

1. Moving character to the right position to match climbing animation and character and platform position. (_climbshift == true). At this moment evenly over a specified time the character is moving so that FrontCheck or FlyClimbCheck position matches with the position of the Grab collider.
2. Moving character from the start position in front of the platform (or under) to the final position - above the platform. When positions FrontCheck and Grab matched moving to the position of ClimbPoint and ClimbPointOnFly starts.

Character's hands have to be directly in the point of clinging to play animation of this state. Also you have to synchronize climbing animation over time and movement.

Project settings

	Configure Layer Collision Matrix

	Configure Input Manager except of default settings.