Extended Character Controller

For Panda3D

Description

The Extended Character Controller for Panda3D is a module that can be used to add a complex player controlled character to your game. This controller module is mainly intended for modern jump and run games but can also be simplified to fit most adventure, Shooter and other kinds of genres.

Features

Basic movements

Walking and running

Basic move towards camera direction

Blend from standing to running and inbetween movement states

Position aware blended rotation toward desired direction

Stepping, supports walking steps up and down (prevents "floating" effect)

Accelerate towards maximum speed

Prevent from running against, head first into walls

Crouching and crawling (Planed but not implemented yet)

Basic crouching and crawling

Stand up limiter

jumping

Basic jumping

Position aware jumping

Jump-enabled tolerance when start falling

Wall runs (Intel Actions)

Running along walls

Edge grab (Intel Actions)

Grabbing edges and moving along them

Climbing (Intel Actions)

Climb ladders and fence-like objects

Moving platforms

Player moves with platform

Player rotates with platform (Around Z-Axis)

Jump forces can be affected by platform movements

Camera

Camera follows player

Collision detection for keeping camera outside of walls and player always in view

Slowly move in place when player is moving

Simple function to toggle between FP and TP view.

Intelligent Actions Further descriptions

The controller can do some special actions dependent on the players actions

Currently there are two intelligent actions available:

Wall Runs:

1) Vertical wall run:

Here the player runs head on towards a wall while pressing the intelligent action key if he's close enough and while pressing the key, the character will start running up the wall or other object which is in front of him.

2) Horizontal wall run:

Similar to the vertical wall run with the difference, that the player needs to run along a wall. If started correctly, the character will run a bit upward and forward on the wall left or right next to him.

Ledge Grab:

If the player is in front of a ledge which he got to by either doing a wall run, jumping or falling the character will initiate a ledge grab. In this mode, the character will hang on the ledge and can be moved left or right along the ledge. He can also move around slight curves but not to step edges. To let go from the ledge grab mode either press Spacebar to climb up on the ledge or 'W' to fall of from the ledge.

Climbing:

If the player is standing next to a climbable area and facing towards it, he will initiate the climbing logic respective to the type of climbable object he is next to. This can either be a horizontal, vertical or both directed climbable area.

How To Use

Integrating this module into your project is as simple as copying the player folder which is located in the src directory into your projects source root directory.

To then use the character import the PlayerController class

```
from player.PlayerController import PlayerController
```

Afterward you can directly set up and run the controller with these few lines.

```
self.p = PlayerController(self.world)
self.p.startPlayer()
```

To set the players initial location and rotation, you can use the following two commands.

```
self.p.setStartPos(LVecBase3(0,0,0))
self.p.setStartHpr(LVecBase3(0,0,0))
```

Whenever you’re done with using the player and like to clean it up, you can use the stopPlayer command.

```
self.p.stopPlayer()
```

In addition to those basic features, there is a list of functions you might want to use.Pause and Resume

The player controller supports pausing and resuming which will stop all features at whichever state they were and respectively resumes them whenever necessary. To pause and resume the player controller use these two commands.

```
self.p.pausePlayer()
self.p.resumePlayer()
```

Change Camera Mode

In addition to pre-setting the camera in the configuration, you can also change the camera mode on the fly between third person and first person with these two calls:

```
self.p.changeCameraSystem("thirdperson")
self.p.changeCameraSystem("firstperson")
```

Core NodePath functions

Some of the basic NodePath functions that apply to the characters model are exposed from within the PlayerController class, which are.

```
show()
hide()
find(searchString)
```

Collision Detection

By default the Panda3D internal physic and collision system is used for the character controller. Note, that the Bullet collision and physics system implementation has been started but not finished yet. At its current state, the Bullet system is not usable. Using it, the following bitmasks (hex values) will be used for the respective tasks:

0x80 -> For event trigger collisions with ghost nodes
0x70 -> For body collisions with walls floors and other objects that the character should not pass through
0x0f -> For ray collisions for example for inteligent movement detection

Make sure to tag your collisions of your levels and objects correctly as otherwise they may not work as expected.

By default the following collision event patterns will be set up and can be listened to from your application logic:

Event containing the from (the character) and into node (other objects) name

In Pattern **%fn-in-%in**
Out Pattern **%fn-out-%in**

e.g.

```
self.accept("CharacterCollisions-out-shopArea", self.disableShopIcon)
```

Event containing just the from (the character) node name

In Pattern **%fn-in**
Out Pattern **%fn-out**

e.g.

```
self.accept("CharacterCollisions-in", self.customCollisionHandler)
```

Configuring the Controller

The character controller comes with an extensive configurability stored within the Config.py script. In there you will need to define the location and names of your characters model and animation, it's size and input mapping. Most other configuration may already fit for most basic setups. Otherwise you can also configure physics, camera settings and various of the character related abilities like jump strength, run speed and so on.

You may also disable the "simple shadow" of the character if you have your own more sophisticated shading system.

Most of the settings in there are documented and should be easy to change to your needs.Demo

For further samples of how to use the controller, a demo.py script is provided in the src folder.

The Demo contains a test level and a fully animated character. It gives you a few options for debugging and testing the character controller. The level contains all possible interact-able objects like ladders and curved climbable areas.

The blender model files for the character and the test level can be found in the doc/srcModels/ folder and can be used as reference.

By default, the following input mapping is applied.

Use Keyboard and Mouse in Third Person view

escape	Exit Application
w, a, s, d	move in the respective direction
spacebar	jump
mouse	move camera
shift	sprint
Home	center camera behind player
del	move camera left
page down	move camera right
end	Move camera down
page up	Move camera up
Left mouse button	Do inteligent action (see Inteligent Actions section)
r	Reset player to start position
p	un-/pause player
f1	Toggle debug information
f2	Toggle between first- and third-person camera mode
f3	Toggle OSD visibility

Note: The player will automatically pause after a given time of idling

Changing the Player Model

Dependend on how many of the features offered by the character controller you need you may need all or just a subset of these model and animation files.

You need one model representing the player. This model must be rigged and have at least the bones “fps_eyes” and “neck” if it should be usable in the first person camera mode.

In addition, the following animation are available to set. Note that some animations are available with a _fp prefix which stands for first person. You can use the same animation for both versions. Usually it's recommended that you lessen the headbob and general head movement in the first person version though to minimize the motion sickness some players might get.

Basic Animations

Third Person Mode	First Person Mode	Description
Idle	Idle_fp	Standing still
Walk	Walk_fp	Slow walking
Run	Run_fp	Normal movement (running)
Sprint	Sprint_fp	Running fast, drains stamina
Jump	Jump_fp	Initiated jump
Fall	Fall_fp	General Falling (also played after jump animation is done)
Landing	Landing_fp	Landing animation played after Fall

Climbing

Third Person	First Person	Description
Climb_Idle		Hanging on the climbable area, no movement
Climb_Up		Climbing ↑
Climb_Up_Left		Climbing ↖
Climb_Up_Right		Climbing ↗
Climb_Left		Climbing ←
Climb_Right		Climbing →
Climb_Down		Climbing ↓
Climb_Down_Left		Climbing ↙
Climb_Down_Right		Climbing ↘
Climb_Exit_Up	Climb_Exit_Up_fp	Climbing up and out of a climbable area

Wall and Ledge Actions

Third Person	First Person	Description
Ledge Grab		
LedgeGrab	LedgeGrab_fp	Hanging on a ledge, not moving
LedgeGrab_Up	LedgeGrab_Up_fp	Climbing up on a ledge

LedgeGrab_Left	LedgeGrab_Left_fp	Moving left while hanging on a ledge
LedgeGrab_Right	LedgeGrab_Right_fp	Moving right while hanging on a ledge
Wall Run		
WallRun_Up	WallRun_Up_fp	Run up vertically on a wall
WallRun_Right	WallRun_Right_fp	Run horizontally on a wall to the right of the player
WallRun_Left	WallRun_Left_fp	Run horizontally on a wall to the left of the player

Code structure

PlayerControler.py

The main class that combines all the functionality

Config.py

This script contains all configurable variables of the player

Camera*.py

This scripts handle the camera control and setup of the character

*FirstPerson = setting up a camera from thew view of the characters eyes with the usual FP setup like mouse controlled view and character rotation

*ThirdPerson = setting up a camera from a shoulder view angle with the usual TP setup like moving the character according to the cameras position

Control.py

This module is responsible for handling the players user input gathered by the respective plugins and also running all the other plugins that are available and active and finally merging all this data together to send it to the desired other parts of the controller like the phyiscs scripts.

InputPlugins/*

In here all the plugins for gathering user input (Keyboard, Gamepads, etc) are located. They all follow a specific design to make the input available to the control part.

ControlPlugins/*

These plugin scripts are responsible for making the character move and behave differently in specific areas as defined within those plugins. All the inteligent actions are coded in here and are called from the control part of the game.

Animator.py

The animator module handles the animation playing of the character and animation blending

Physics*.py

This contains the physics interface of the character. Everything related to physics and collision detection should go here to make it simply interchangeable with other physic systems

*Internal = using P3D's internal physic and collision system

*Bullet = using Bullet physic and collision system

Limitations

Physics and collisions

Currently only the Panda3D internal physics and collision system is truly implemented and works out of the box. A lot of work went into performance optimization and stabilizing the code behind it to make it work with large scale applications.

A Bullet system integration is work in progress but currently not usable due to lack of knowledge and volunteers working on it.