The KAM3RA 3rd--Person camera system, written in fully-commented C# for Unity Free and Pro, is professional-level camera control logic similar to 3rd-person cameras used in typical MMO and RPG games. Designed specifically for 3rd-Person camera control, the system uses Unity's Rigidbody physics and reliable camera collision to control your main character while handling translation and orientation of the camera. Fully configurable, the API features extensible, easy-to-grasp logic and full documentation. An full-featured demo scene is included.

Demo Scene

The demo scene is located at **/DEMO/!Scene**. The demo uses free Unity assets as well as some additional assets provided by Regress Software, which are free for your use. It's recommended that you define Walk, Strafe and AutoRun keys in your project's input definitions (**Edit -> Project Settings -> Input**), however this is not required. Defaults are:

• MOVE:	W, S, UP, DOWN
• TURN:	A, D, LEFT, RIGHT
• STRAFE:	Q,E OR A,D + RIGHT MOUSE
• JUMP:	SPACE
• WALK:	SHIFT
• TOGGLE RUN:	PAGE UP
• CAMERA LOOK:	RIGHT MOUSE
• SELECT NPC:	LEFT CLICK NPC
• ENTER VEHICLE:	RIGHT CLICK VEHICLE
• EXIT VEHICLE:	X
• RESET VEHICLE:	R
• TOGGLE NAMES:	N

There are several classes that support the demo scene. Feel free to use these classes in your projects, however please note they're for demo purposes and Regress Software does not support them.

Class File Name	Description	Custom Inspector
DEMO/Classes/**Game**.cs	Demo class for user interaction with the demo scene.	N/A
DEMO/Classes/**Player**.cs	Demo class for a main character/actor with sound effect for footsteps.	DEMO/Editor/**PlayerInspector**.cs
DEMO/Classes/**NPC**.cs	Demo class for NPC movement and behavior.	DEMO/Editor/**NPCInspector**.cs
DEMO/Classes/**Kart**.cs	Demo class for simple for a Mario-Kart-style vehicle.	DEMO/Editor/**KartInspector**.cs
DEMO/Classes/**Flyer**.cs	Demo class for an MMO-style flying vehicle.	DEMO/Editor/**FlyerInspector**.cs
DEMO/Classes/**Props**.cs	Demo helper class for miscellaneous objects.	N/A
DEMO/Classes/**Prop**.cs	Demo helper class for miscellaneous objects.	N/A
DEMO/Classes/**Sundome**.cs	Demo class for simple sphere-based Sundome.	N/A

KAM3RA Namespace

The KAM3RA namespace is comprised of three classes and two editor classes.

Class File Name	Description	Custom Inspector
KAM3RA/Classes/**Actor**.cs	The game object being controlled by the User class. Usually this is your main character or other in-game character.	KAM3RA/Editor/**ActorInspector**.cs
KAM3RA/Classes/**User**.cs	The input controller for processing mouse and key input, computing 2D velocities and passing them to an Actor. Must be attached to a camera, usually the main camera.	KAM3RA/Editor/**UserInspector**.cs
KAM3RA/Classes/**Range**.cs	Utility class, useful for easily storing min/max values.	N/A

How to Use KAM3RA

Using KAM3RA is relatively straightforward. The system is fairly straightforward and does not require much effort to get up-and-running.

1. Attach a User script to your main camera:

    - **Sensitivity**: When sending input (e.g. via keys or mouse), how fast you look and turn with the camera.
    - **Damping**: When *not* sending input, how fast the look and turn amounts converge to zero.
    - **Look Range**: The minimum and maximum angles when looking up and down.
    - **Zoom Range**: The minimum and maximum distance when using the scroll wheel.
    - **Mouse Button Control**: Which mouse buttons control the Actor and the Camera, respectively. Selections are Left, Right, Middle, None (free movement without using a button) and Disabled (has no effect).
    - **Max Mouse Velocity**: The maximum amount of momentum that can build up when using the mouse.
    - **Align**: Click this button to align the Scene view in Unity with the camera. This does the same thing as Game Object, Align View to Selected from Unity's main menu.

2. Attach an Actor script to the game object, usually your main character, you wish to control:

    - **Type**: Choose Ground for a non-flying character, Fly for a simply gravity-enabled flying object, or Hover for non-gravity-enabled flight.
    - **Player**: Check this to tell the User class that you want it to control this character. **Note that Player is false by default.**
    - **Drag**: The drag when moving and colliding.
    - **Max Speed**: The maximum speed at which your character will attempt to move.
    - **Acceleration**: Expressed as a percentage of Max Speed, how quickly your character will achieve Max Speed.
    - **Momentum**: How slowly your character comes to a stop. Zero is a dead stop.

- **Jump Height**: The approximate height your character can jump.
- **Eye Height**: Where the camera is looking through your character. Zero to One is from feet (presumed to be the model's local position) to the top of the model's head.
- **Max Slope**: When colliding, the maximum slope angle at which your character can move.
- **Animation Map**: This section maps animation names to strings that are used in the Actor class to get and set state. There are 12 default states that the Actor class uses:

    1. Idle
    2. WalkForward
    3. WalkBack
    4. RunForward
    5. RunBack
    6. TurnLeft
    7. TurnRight
    8. StrafeLeft
    9. StrafeRight
    10. Jump
    11. Fall
    12. Fly

To add a new state, type in the State and its Name (in orange at the bottom) and press the + button. To delete a state, press its – button. **Warning: Deleting any of the 12 default states may cause problems – don't do that unless you know what you're doing!**

Please read code comments for specific details on KAM3RA's variables and methods.

FAQ

Question	Answer
What is KAM3RA?	KAM3RA is a Unity asset for adding 3rd-Person camera functionality to your Unity game or simulation.
There are other 3rd-person cameras in the Asset Store, including a demo from the Unity team. Why not use one of those?	While Unity and individual programmers have written some great 3rd-person code, it always seemed to be lacking in some way or another for us. Whether missing a crucial feature, no Rigidbody support or thorny to use and extend, we weren't satisfied. So we dove in and created our own Rigidbody-friendly 3rd-person camera system.
Does KAM3RA use Rigidbodies?	Yes! On startup, the Player class will even create and configure CapsuleCollider and Rigidbody objects for you.
What flavor of Unity is KAM3RA?	KAM3RA is 100% C#, fully commented and carefully constructed so that it's readable, understandable and extensible.
Does KAM3RA include a demo scene?	KAM3RA includes a large demo scene with a main character, several AI-controlled characters, Unity terrain, trees and other models for testing movement and physics. We even threw in an old version of our day-night cycle code, just for fun and ambience.
May I use KAM3RA demo assets in my Unity game?	Most of the assets are Unity assets and fall under Unity's licensing. You're also welcome to use any of the other assets in the demo, although these are low-poly "programmer art" pieces that would have us scratching our heads if you wanted to actually use them in a professional title.
What are the recommended Unity physics settings for KAM3RA?	We recommend using a custom physics material with less friction and more gravity. This material is included in the package.
Who wrote KAM3RA?	KAM3RA was written by Regress Software, a small group of game developers who have been making games for over 15 years.

Release Notes

Version	Notes
1	Initial release.

Support

Contact kam3ra@regress.com, or visit the website at www.regress.com/kam3ra, with questions and/or bugs.