Введение в Blender Game Engine (BGE). Урок 1

1. Что такое BGE

Уникальной особенностью Blender является встроенный в него так называемый "игровой движок" (англ. game engine). Это программный компонент, который позволяет создавать небольшие игры, не выходя из среды трехмерного моделирования Blender.

Game engine:

1. придает объектам "физические" свойства; объекты начинают вести себя так, как в реальном мире: падают под действием силы тяжести, сталкиваются и т.д. (возможно следует отметить, что в этом смысле движок еще "сыроват");
2. позволяет наблюдать поведение объектов в реальном времени и воздействовать на них;
3. записывает происходящее в режиме игры в анимацию;
4. позволяет визуально настраивать (программировать) поведение и взаимодействие объектов;
5. а также встраивать скрипты на языке программирования Python.

Режим игры – это режим действия игрового движка. Он запускается при нажатии клавиши P на клавиатуре. Выход из него – клавиша Esc. Лучше перед запуском игры переключиться в режим отображения текстур (Alt+Z).

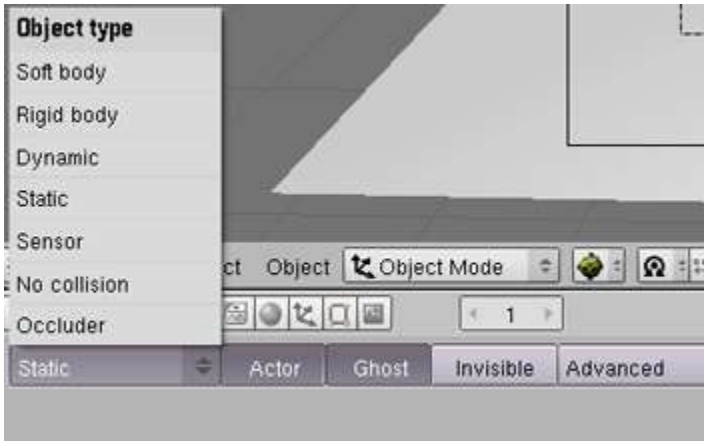
Задание. *Расположите на сцене плоскость и любой другой меш-объект над ней. Включите режим отображения текстур, затем режим игры. После выйдите из режима игры.*

Для настройки поведения объектов для режима игры предназначена панель кнопок Logic (F4). На этом уроке мы рассмотрим ее левую часть – настойки объекта.

2. Типы объектов

В реальном мире объекты сделаны из различных материалов и, естественно, обладают различными свойствами. Игровой движок (BGE), конечно, не позволяет симулировать такое разнообразие, но кое-что может. К тому же, BGE позволяет манипулировать и "нефизическими объектами", т.е. теми для которых как бы нет влияющего окружающего мира (в реальном мире таких объектов не существует).

Тип выбранного объекта (например, куба) задается с помощью меню Object Type.



Occluder (occlude – закрывать, преграждать). В определенных условиях увеличивает производительность путем исключения объектов, расположенных за объектом-«окклюдером». Уместно использовать в сложных сценах.

No collision (collision - столкновение). Отключает у объекта возможность столкновения с другими объектами. Т.е. если dynamic-объект встретит на своем пути no_collision-объект, то пройдет сквозь него, как будто его нет. Рекомендуется использовать для объектов, которые никогда ни с чем не столкнуться: выбор этого типа объекта увеличивает производительность игры.

Sensor (датчик). Предназначен для более сложного управления столкновениями.

Static (статический, неподвижный). Такой тип объекта может участвовать во взаимодействиях с физическими объектами (играть роль препятствий), хотя на него не влияют никакие виртуальные физические силы. В режиме игры он может быть перемещен за счет изменения своего положения, но не под влиянием сил. Два static-объекта при пересечении их путей не сталкиваются, а проходят сквозь друг друга.

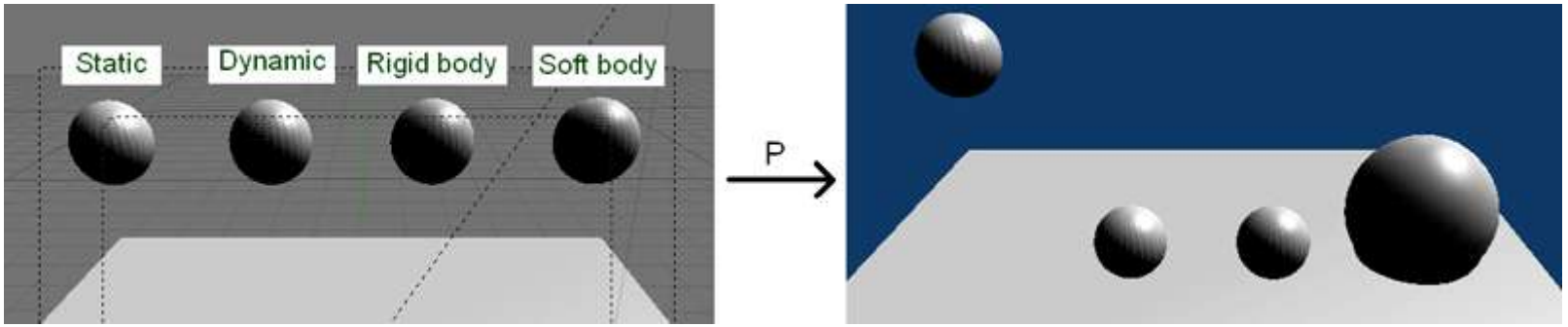
Dynamic (динамический). Объект, на который "действуют" законы физики: сила тяжести, масса, воздействие внешних сил, ускорение и др.

Rigid body (твердое тело). Более реально ведущий себя по сравнению с Dynamic "физический" объект.

Soft body (мягкое тело). "Физический" объект, который ведет себя подобно резиновому мячу или мягкой игрушке, т.е деформируется (изменяет свою форму) при столкновениях с другими объектами.

Опыт 1. **Типы объектов**

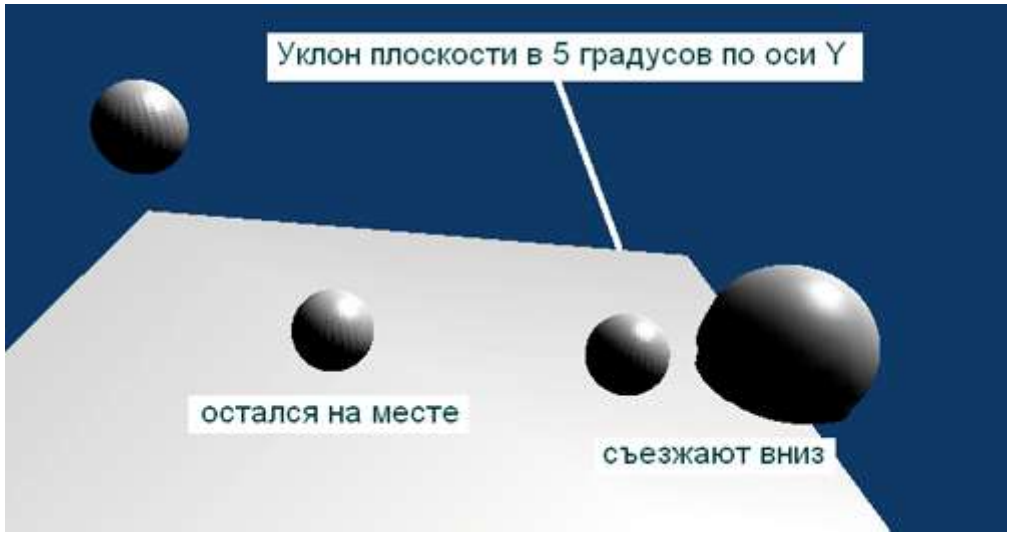
Для того, чтобы сразу протестировать как ведут себя некоторые типы объектов в режиме игры, расположим несколько сфер над плоскостью и нажмем P. Вот что мы увидим по прошествии нескольких секунд:



Видно, что с объектов типа Static ничего не произошло, Dynamic, Rigid Body и Soft Body упали под действием виртуальной силы тяжести. Причем Soft Body упал и слегка «размазался», как и положено мягкому телу, но почему-то меш-объект остался прежнего размера (это недоработка Blender или как-нибудь настраивается отдельно? Примечание: если при создании сферы сразу указать небольшой размер, то она будет мелкой при создании из нее мягкого тела).

Опыт 2. **Dynamic и Rigid body**

Чтобы заметить разницу между Rigid body и Dynamic, развернем плоскость на 50 по оси Y или X. Теперь можно наблюдать, что при падении Rigid Body продолжает свое движение как естественный объект (также ведет себя и Soft Body), а Dynamic остается спокойно лежать на плоскости. Конечно, если наклон плоскости сделать больше, то вниз покатится и динамический объект.



3. Настройки

В предыдущем опыте можно заметить, что при смене типа объектов меняются кнопки настоек ниже. Здесь основная часть настроек свойств объекта. Рассмотрим некоторые.

Общие:

Actor (актер). Для версии 2.49 осталось непонятным, на что влияет данных параметр.
Ghost (призрак). Объект будет проходить сквозь другие объекты при столкновении с ними, при этом никак на них не влияя.
Invisible (невидимый). Объект становится невидимым, но присутствует на сцене. Объект-невидимка может взаимодействовать с другими объектами.
Mass (масса). Виртуальная масса объекта. Настройка данного параметра имеет большое значение для "физических" объектов при воздействии на них той или иной силы. Чем больше масса, тем больше нужно приложить силы для ее перемещения.

Rigid body:

Radius (радиус). Влияет на площадь взаимодействия с объектом. Радиус виден как розовая пунктирная окружность в 3D окне. Если не виден, то увеличьте значение. Фактически границы окружности и определяют размер взаимодействующей области (объекта).
No sleeping (не спящий). При включении-отключении данной опции в версии 2.49 изменений в поведении объекта при применении к нему силы замечено не было.
Остальные настройки пока опустим, т.к. они достаточно специфичны и требуют начальных знаний физики.

Soft body:

Shape Match (shape – форма, match – выравнивать, противостоять и др.). Если кнопка будет выключена, то объект не сохранит свою форму, а сложится как тряпка (при достаточном количестве вершин).

Bending Const (bending – сгибание, constraints – ограничения). Если отключить данную кнопку, то объект при столкновении будет сильнее изменять свою форму.
kMT. Влияет на упругость объекта, связана с Shape Match. Увеличив данное значение, например, у сферы, вы получите объект поведением похожий на мяч.

4. Практическая работа

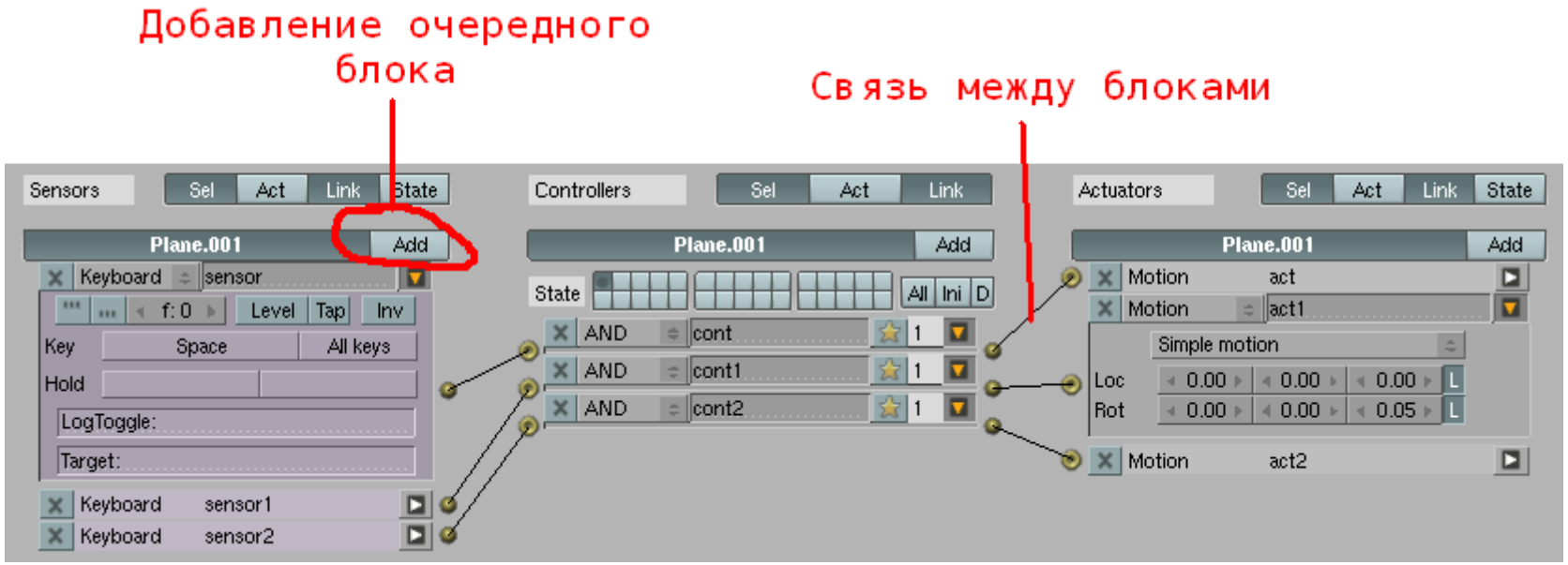
1. Разместите на сцене плоскость, увеличьте ее. Над плоскостью друг над другом расположите три одинаковых куба, каждый куб должен быть объектом Rigid body. Запустив режим игры, отметьте, как упали кубики.
2. У среднего куба увеличьте Radius до значения 1.5. Снова запустите игру и отметьте изменения.
3. Включите для среднего куба кнопку Invisible и снова запустите режим игры. Участвует ли невидимый куб во взаимодействии объектов?
4. Удалите со сцены все три куба. Добавьте сферу с 16-тью сегментами и кольцами (уменьшение частей меш-объектов влияет на производительность),
   расположив ее над плоскостью. Сделайте ее объектом Soft body и продублируйте по оси X четыре раза (все пять объектов должны быть над плоскостью,
   увеличьте плоскость при необходимости). Желательно развернуть камеру таким образом, чтобы она "смотрела" прямо на ось X.
5. Для первого объекта Soft body оставьте значения настроек по умолчанию. У второго – увеличьте значение kMT почти до единицы. У третьего – отключите Shape Match, у четвертого – Bending Const, а у пятого – обе кнопки.
6. Запустите режим игры. Отметьте изменения каждой из сфер. Если разница между первой и четвертой сферой? В чем она заключается?

Сенсоры, контроллеры и активаторы. Урок 2

Уроки по Blender Game Engine

Панель Logic (F4) можно условно разбить на четыре части настроек: свойства объекта (рассмотренные на прошлом уроке), сенсоры, контроллеры и активаторы. **Сенсоры** объекта позволяют ему воспринимать различные внешние «раздражители», такие как нажатие клавиши, движение мыши, касание другого объекта и др. **Контроллеры** в основном предназначены для связывания определенным образом сенсоров и активаторов. **Активаторы** могут выполнять различные действия над данным объектом или другими. Некоторые настройки активаторов могут различаться у разных типов объектов.

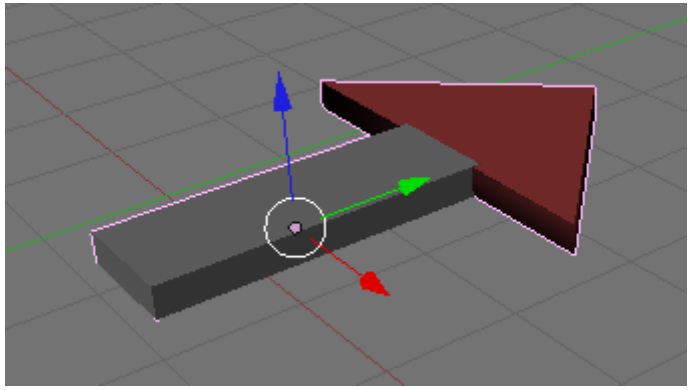
Для того чтобы добавить очередной сенсор, контроллер и активатор нужно нажать на кнопку **ADD**. Связь между ними обозначается линиями, которые создаются, если зажав ЛК мыши потянуть ей от одного узелка к другому (линию можно удалить, если навести мышь и нажать клавишу Del). У объекта может быть множество групп сенсоров, контроллеров и активаторов.



На этом уроке будет описано, как управлять объектом **Static** с помощью клавиатуры. Роль объекта сыграет стрелка, поворачивающаяся по двум осям и двигающаяся указателем вперед.

1. Создание объекта

Сначала создадим объект. Можно использовать любой меш-объект, но желательно, чтобы он был ассиметричен, т.к. в этом случае легче будет наблюдать его повороты.



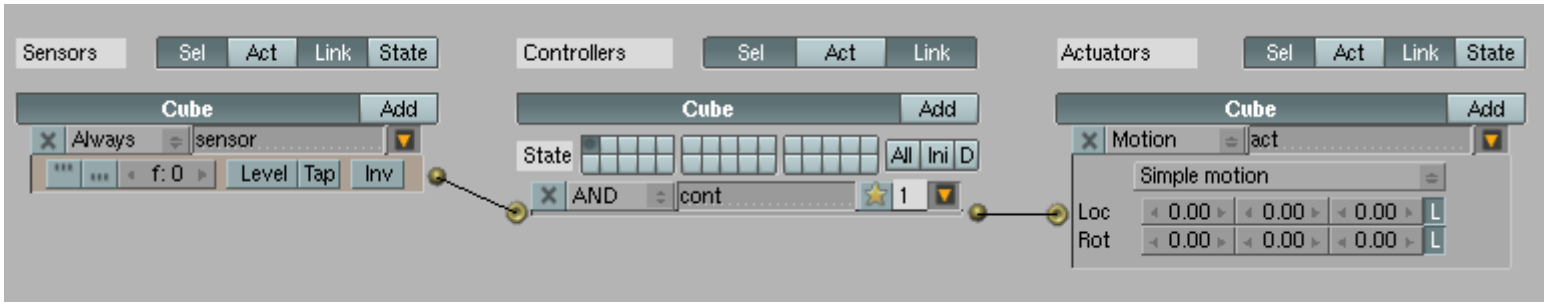
Такую стрелку можно получить множеством способов. Ниже описан один из них:

1. Вид сверху.
2. Добавить плоскость.
3. Перейти в режим редактирования. Удалить нижнюю правую вершину. Выделить три оставшиеся, нажать F (получиться треугольная плоскость). Далее, не снимая выделения, повернуть плоскость по оси Z на 450, таким образом, чтобы будущий указатель «смотрел» вверх (вдоль оси Y).
4. Не выходя из режима редактирования, добавить еще одну плоскость ниже. Сжать ее по оси X, переместить под указатель; – это ножка стрелки.
5. Вид из камеры.
6. Выбрать режим выделения граней и выдавить (экструдировать), тем самым придав стрелке объем.
7. Выйти из режима редактирования.

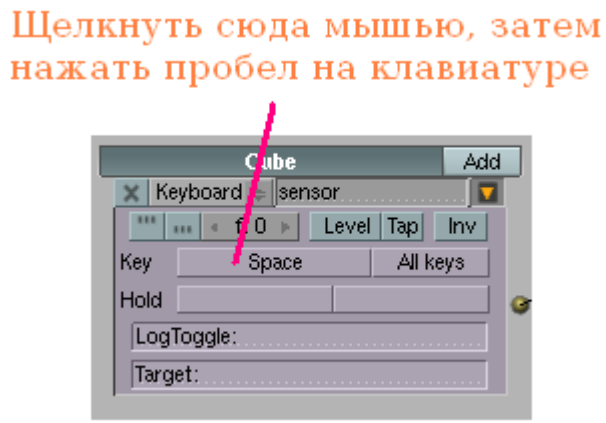
2. Движение вперед

Заставим нашу стрелку двигаться указателем вперед при нажатии на клавишу *Пробел*.

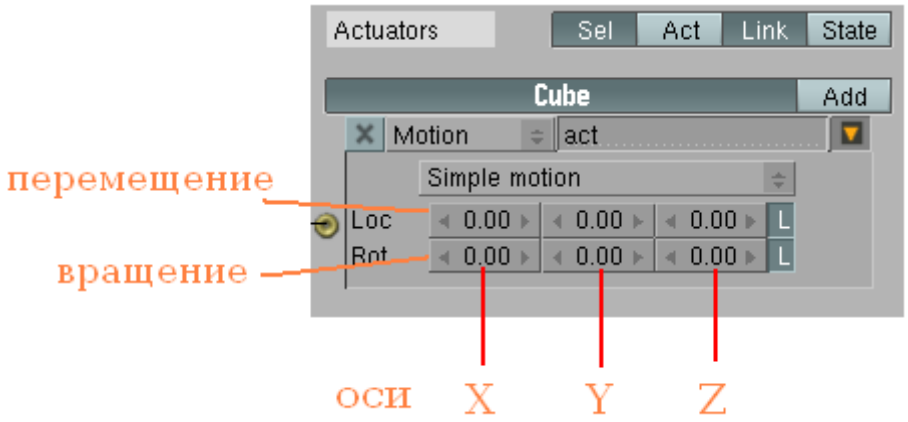
Выделив стрелку (которая должна быть static-объектом), добавьте сенсор, контроллер и активатор и соедините их между собой.



По умолчанию в качестве сенсора выступает **Always** (всегда). Такой сенсор будет действовать постоянно. Нам это не подходит. В выпадающем списке выбираем **Keyboard** (клавиатура). Нажимаем в поле **Key** мышью, затем нажимаем клавишу *Пробел* (Space).



Теперь настроим активатор. Активаторы бывают разные (это на будущее). По умолчанию стоит **Motion** (движение). В нижней части есть две строки: **Loc** (location – положение) и **Rot** (rotation – вращение). В каждой строке по три поля: в них задаются величины изменений по оси X, Y или Z при воздействии.

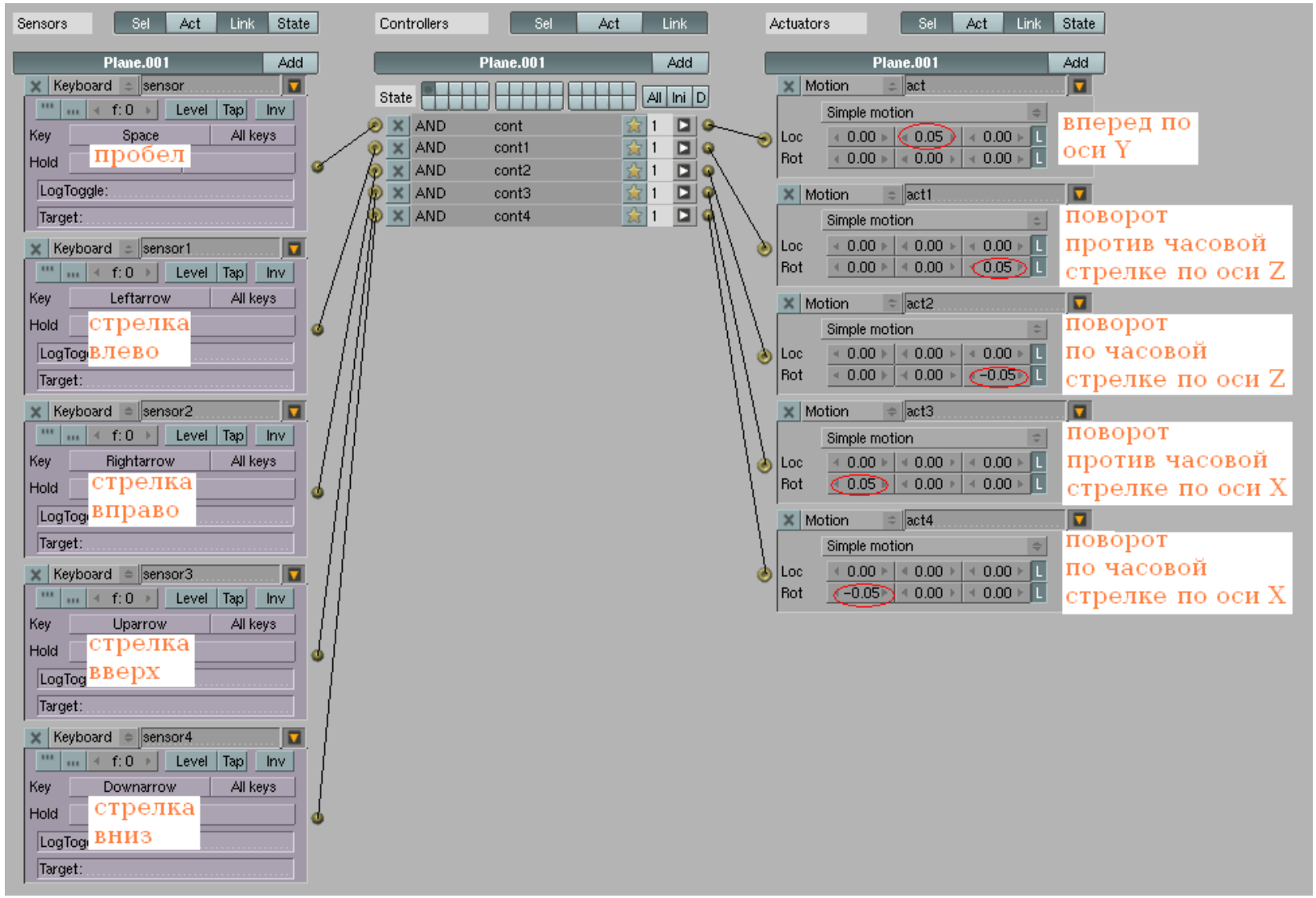


Поскольку наша стрелка должна двигаться вперед и для нее это ось Y, то следует изменить значение в первой строке во втором столбце. Щелкните по нулям там и пропишите значение в 0.05.

Теперь при запуске режима игры (P) и нажатии на клавишу пробел стрелка будет двигаться вперед.

3. Повороты

Для того, чтобы стрелка при движении могла оказаться в любой точке трехмерного пространства достаточно настроить ее поворот только по двум осям: X (повороты вверх и вниз) и Z (налево и направо). Поворачиваться стрелка будет при нажатии на стрелки на клавиатуре. Требуется добавить еще четыре группы сенсоров, контроллеров и активаторов.



Теперь, запустив режим игры, можно передвинуть стрелку в любое место 3D-окна, предварительно повернув ее с помощью стрелок на клавиатуре.

4. Локальные и глобальные оси

В конце строк настроек **Loc** и **Rot** есть переключатели **L**. По умолчанию эти кнопки нажаты и обозначают локальную привязку осей. Это означает, что оси X, Y и Z как бы находятся внутри объекта и при его повороте поворачиваются вместе с ним. Поэтому-то у нас стрелка все время двигается указателем вперед, как ее не поверни.

Если выключить кнопку **L** напротив **Loc** в блоке, отвечающем за движение по оси Y, то стрелка будет двигаться только в одном направлении (по глобальной оси Y) не зависимо от поворота своего указателя. Глобальные оси в 3D-окне обозначаются красной (X), зеленой (Y) и синей (Z) линиями.

5. Практическая работа

1. Создайте модель стрелки и задайте ей возможность движения и поворота с помощью Blender Game Engine так, как описано в уроке.
2. Исследуйте влияние на движение стрелки включенной и отключенной Local Transformation (кнопка **L**).
3. Добавьте на сцену плоскость и пару других меш-объектов, расположив их над плоскостью. Исключая плоскость и стрелку, задайте для объектов тип **Rigid body**. Запустите режим игры и попробуйте столкнуть объекты с плоскости с помощью стрелки.