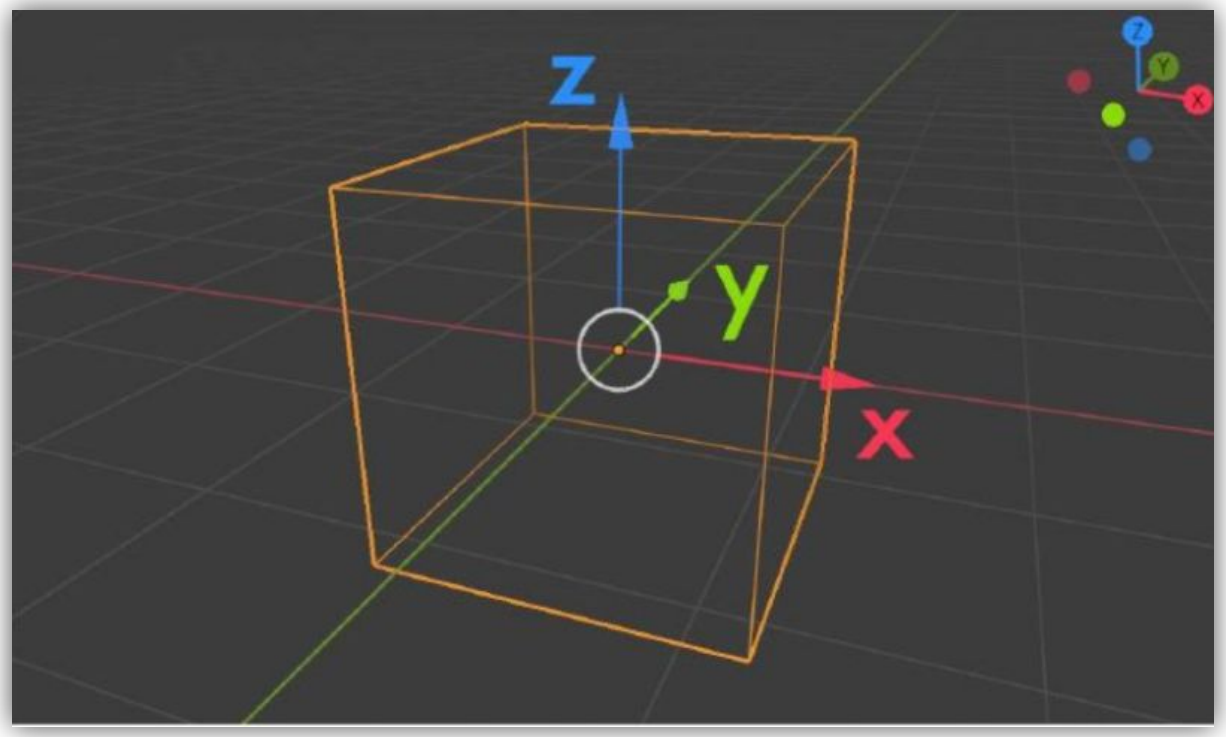
Разработка игры в жанре песочница.

Panda3D

Игровой движок, включающий графику, звук, ввод-вывод, обнаружение столкновений и другие функции, относящиеся к созданию 3D игр.

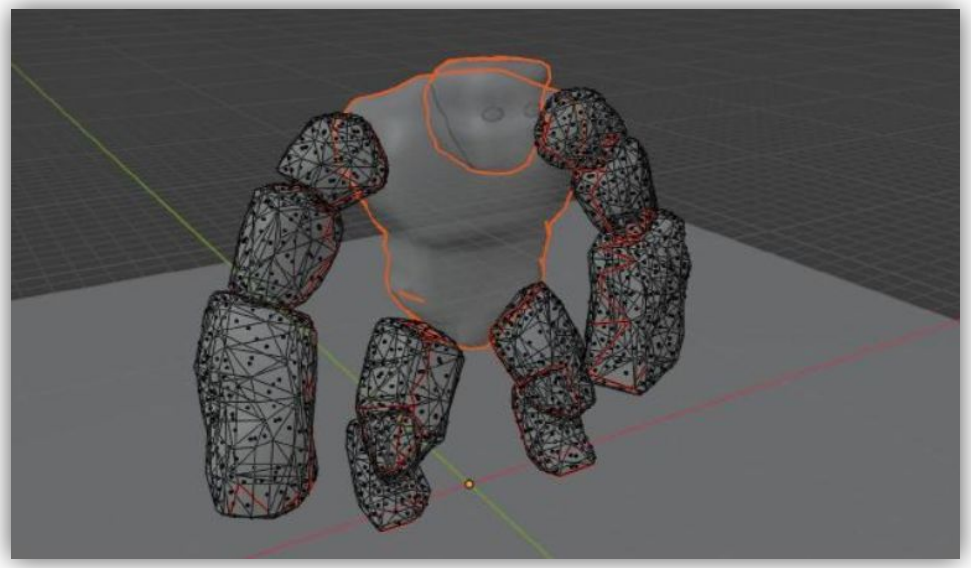
В Panda3D плоскость **XY** представляет поверхность земли, ось **X** направлена вправо, ось **Y** – вперёд, а ось **Z** – вверх.



Шаги получения трёхмерного изображения:

1.Моделирование – создание математической модели сцены и размещение в ней объектов.

2.Рендеринг – построение геометрической проекции трёхмерной модели сцены на экране.



Сцена – виртуальное пространство моделирования. Она включает в себя следующие объекты.

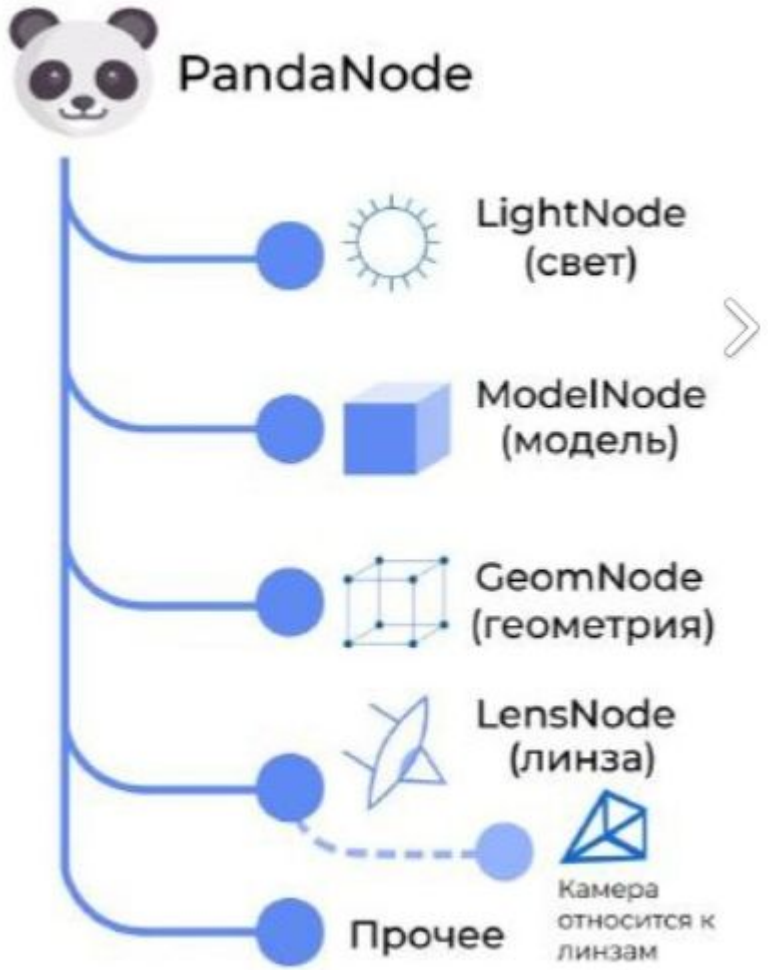
- **Геометрия** – построенная с помощью различных техник модель, например – здание;
- **Материалы** – информация о визуальных свойствах модели;
- **Источники света** – настройки направления, мощности, спектра освещения;
- **Виртуальные камеры** – выбор точки и угла построения проекции.

Основы Panda3D

Метод run – содержит главный цикл движка Panda3D. В нём рисуется кадр и выполняются фоновые задачи. Этот метод вызывается единожды в конце программы.

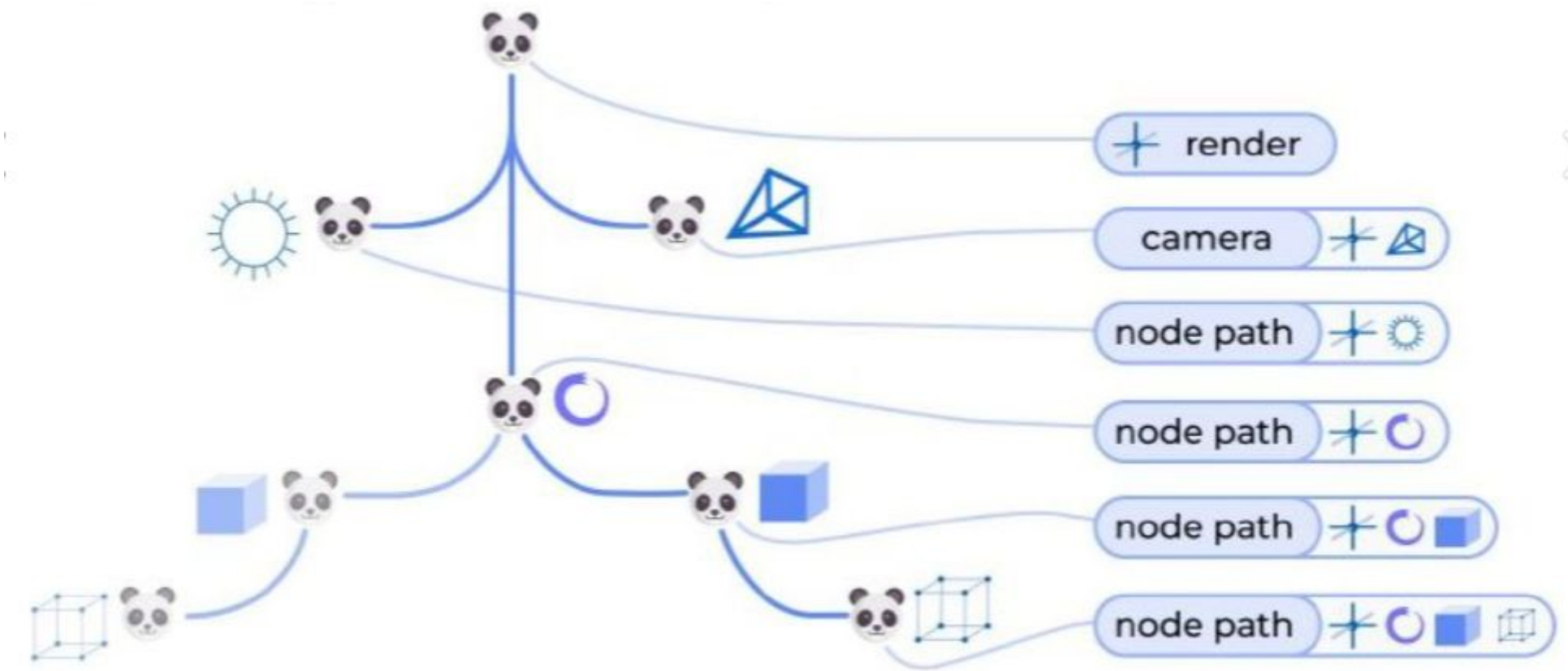
Глобальные переменные Panda3D
Loader – загрузчик, используемый для загрузки различных типов объектов.
Render – вершина текущей 3D сцены, которая отображается на экране.

Метод **model = loader.loadModel()** загружает указанный файл модели. Возвращаемое значение – **NodePath**, «узел», контейнер объектов Panda3D, который содержит загруженную модель.



Все объекты движка Panda3D находятся в узлах графа сцены.

Граф сцены – это дерево, содержащее все модели, которые необходимо выводить на экран. Корень этого дерева – **render**.

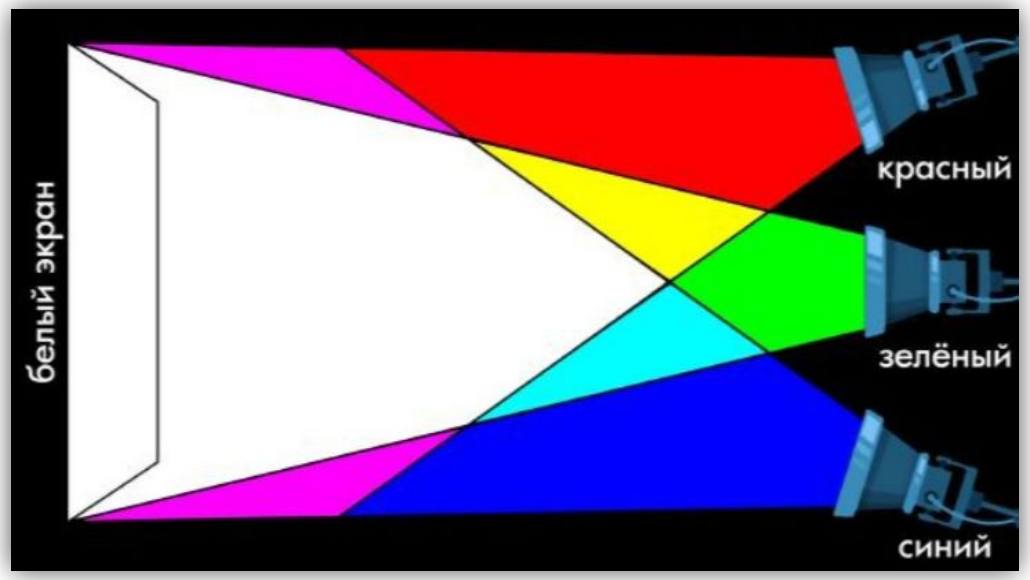


Объекты не помещённые в граф сцены не будут выведены на экран. Одно из наиболее частых действий с графом сцены – это смена родительских узлов.

Методом **model.reparentTo(render)** мы устанавливаем родителя для нашего объекта, тем самым помещая его в граф сцены, что делает его видимым.

Цветовая модель RGBA

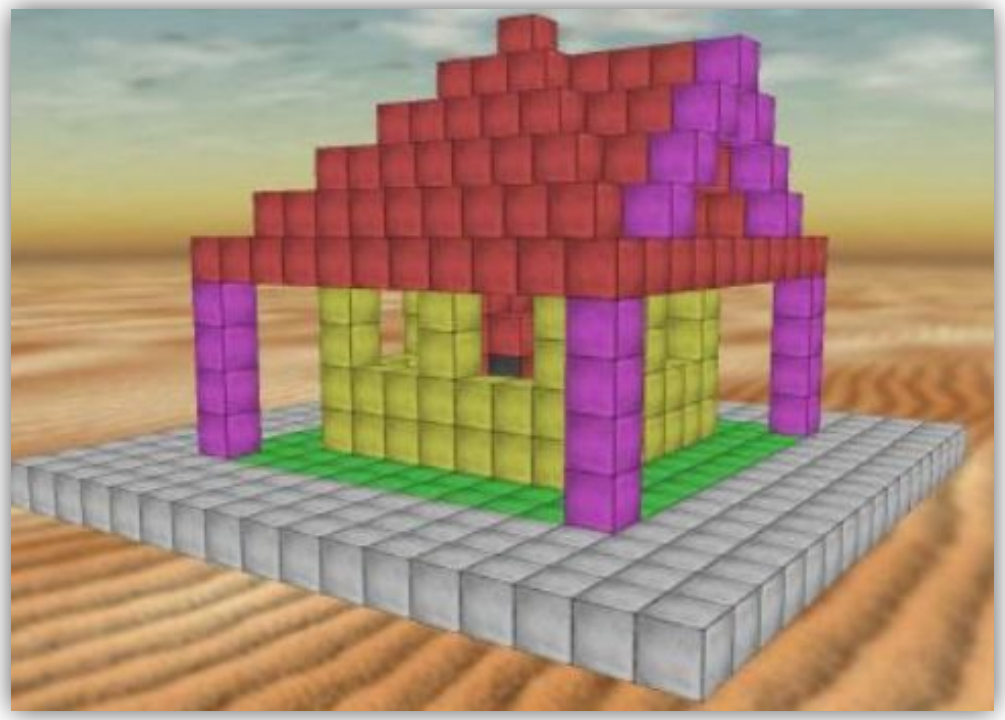
В Panda3D используется цветовая модель **RGBA**. Она названа так по трём заглавным буквам названий цветов, лежащих в её основе: **Red**, **Green**, **Blue**. Если смешивать эти цвета в различных сочетаниях то можно получить все цвета радуги. Буква **A** происходит от названия параметра **прозрачности Alpha**. Он определяет общую прозрачность цвета.



Вложенные списки и словари

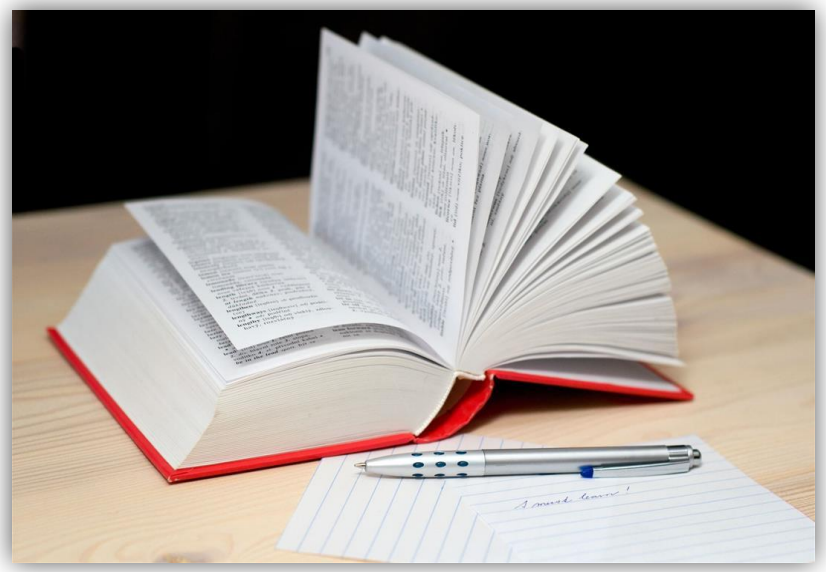
Примеры вложенных списков:

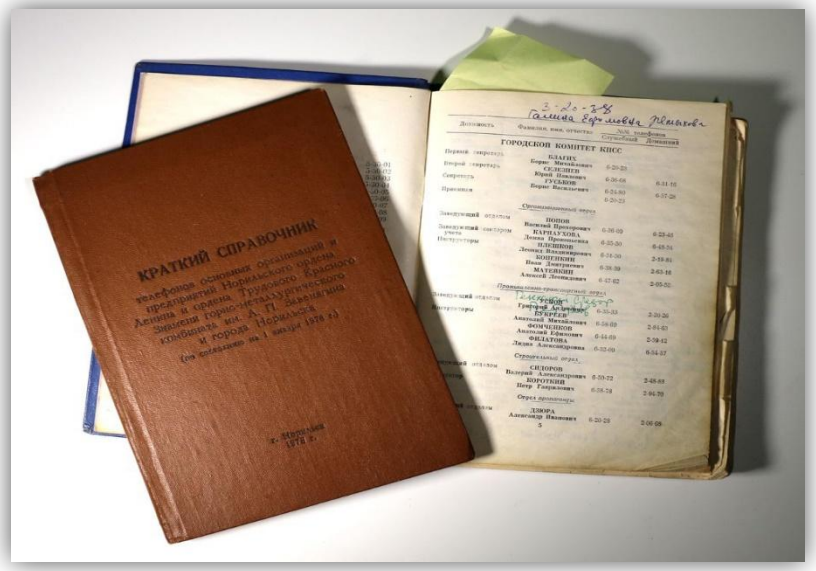
- Двумерный список – **бланки, Excel-таблицы**
- Трёхмерный список – **карта игры**



Пример словарей: привычные **бумажные словари**. В них **ключом** является **слово-заголовок статьи**, а **значением – сама статья**. Для того, чтобы получить доступ к статье, необходимо указать слово-ключ.

Другой пример: **телефонный справочник**. В нём **ключом** является **имя** или **название**, а **значением – номер телефона**. И словарь, и телефонный справочник хранятся так, что легко найти элемент словаря по известному ключу.

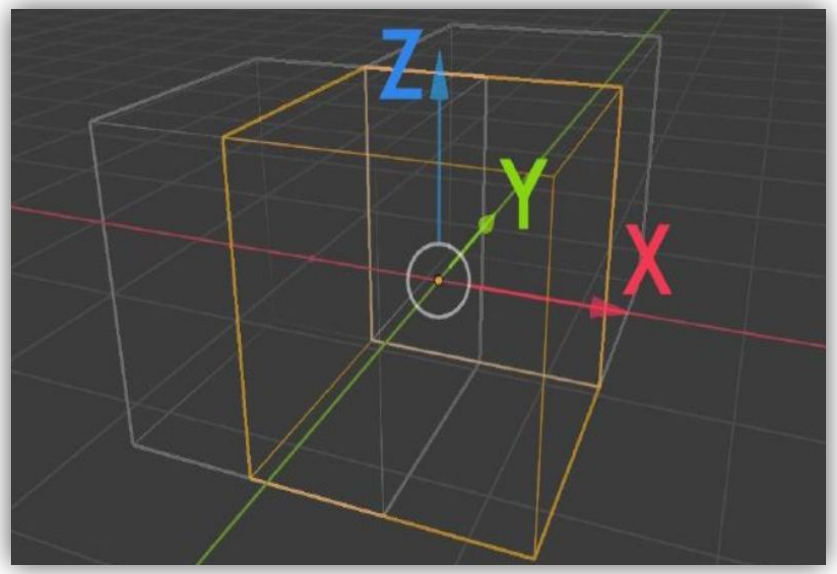


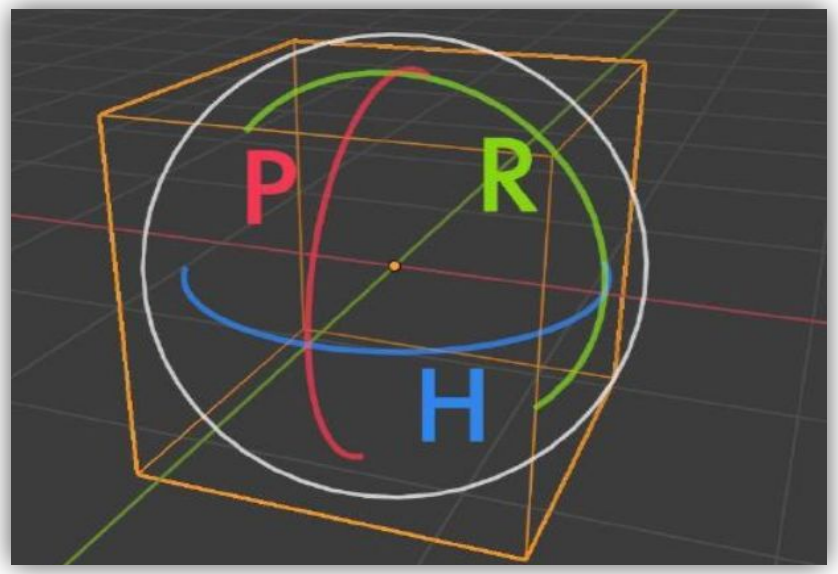


Ориентация в 3D пространстве

В Panda3D объекты можно **перемещать** вдоль трёх осей: по **X – вправо и влево**, по **Y – вперёд и назад** и по **Z – вверх и вниз**.

Также объекты можно **вращать** вокруг трёх осей.





Объект **base.camera** – **камера**, используемая для рендеринга трёхмерного графа сцены. Мы можем применять к этому объекту методы изменения положения и ориентации для управления камерой в игре. К примеру для начала необходимо вызвать метод **base.disableMouse()** для того, чтобы отключить управление камерой по умолчанию.

Обработка событий

Обработка событий

События происходят, когда пользователь что-то делает.
Например, *щёлкает мышкой* или *нажимает клавишу*.

Когда происходит событие, **менеджер событий** Panda3D проверяет, написали ли мы **функцию обработчика событий**.
Если это так, будет вызван обработчик событий.

Если мы хотим, чтобы наше приложение закрывалось по **нажатию клавиши «Esc»**, то мы можем зарегистрировать следующее событие:

Base.accept(“escape”, base.userExit)

Метод userExit **закрывает приложение** Panda 3D

Менеджер задач

Задачи – это специальные функции, которые вызываются с заданной периодичностью во время выполнения приложения.

Например, мы можем написать функцию **myTask**, в которой будем отслеживать изменение положения мышки и соответствующе поворачивать камеру. После этого необходимо регулярно вызывать её, как задачу.

Имя модуля

Многие большие проекты состоят из нескольких основных классов, каждый из которых реализует свой особый функционал.

В ООП принято каждый такой класс помещать в отдельный **модуль**.

- Это позволяет разрабатывать каждый класс отдельно от других
- Упрощает отладку каждого класса

Чтобы использовать класс в других программах, необходимо его импортировать, однако необходимо помнить о том, **что при импорте модуля содержащийся в нём код исполняется**.

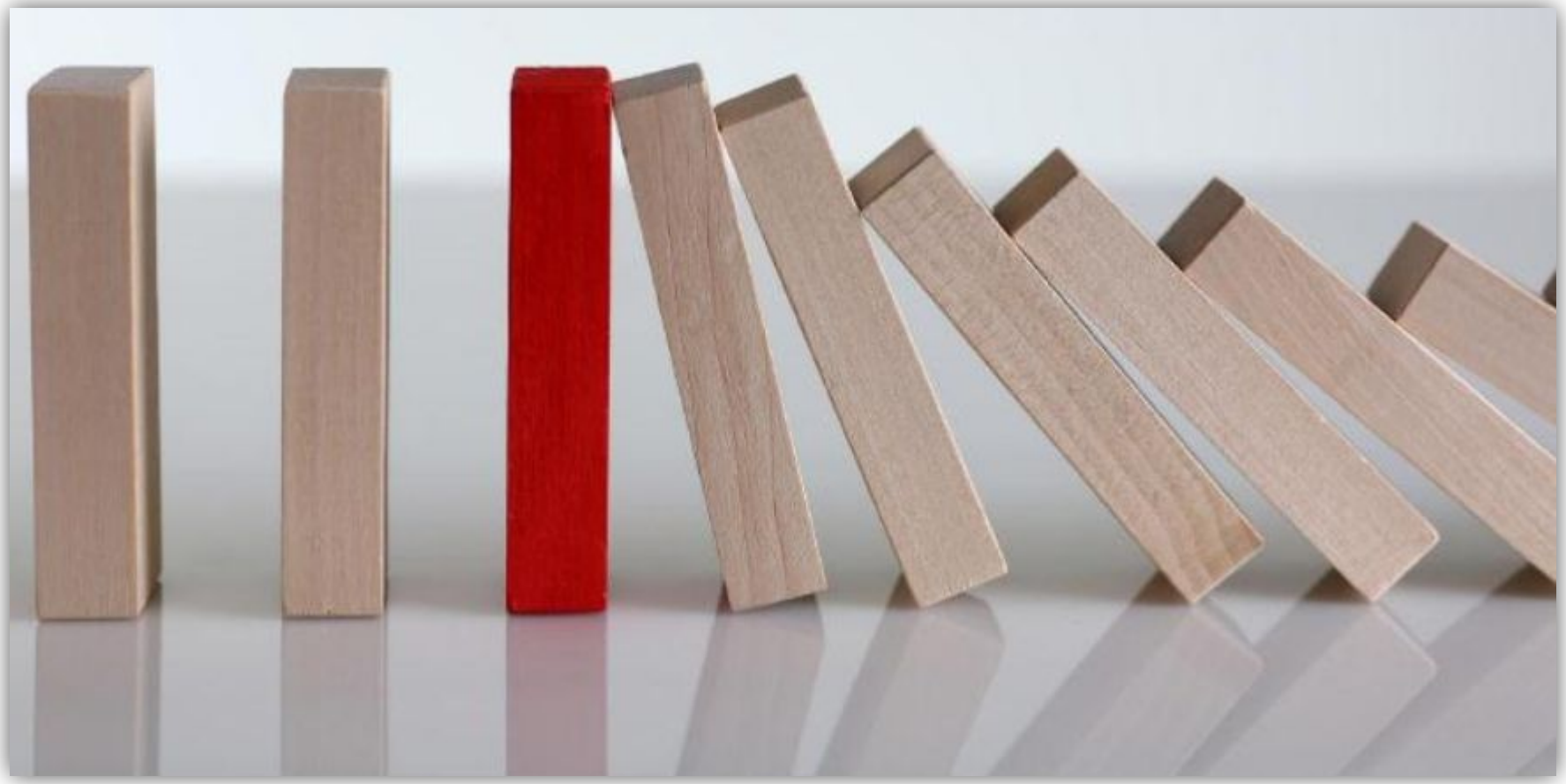
Свойства класса и объекта

Свойства класса принадлежат **классу**. Доступ к ним могут получать все объекты этого класса. Свойство класса существует только **одно**, поэтому когда любой из объектов изменяет свойство класса, это изменение **отразится и во всех остальных объектах того же класса**.

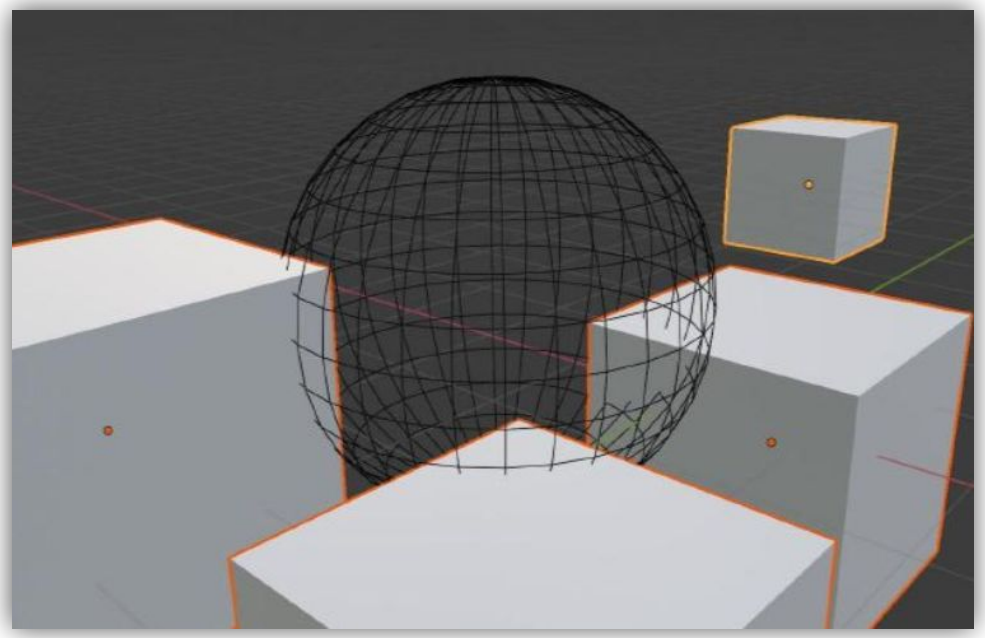
Свойства объекта принадлежат **каждому отдельному объекту** класса. В этом случае у каждого объекта есть своя **собственная копия свойства**, не разделяемая с другими такими же свойствами в других объектах

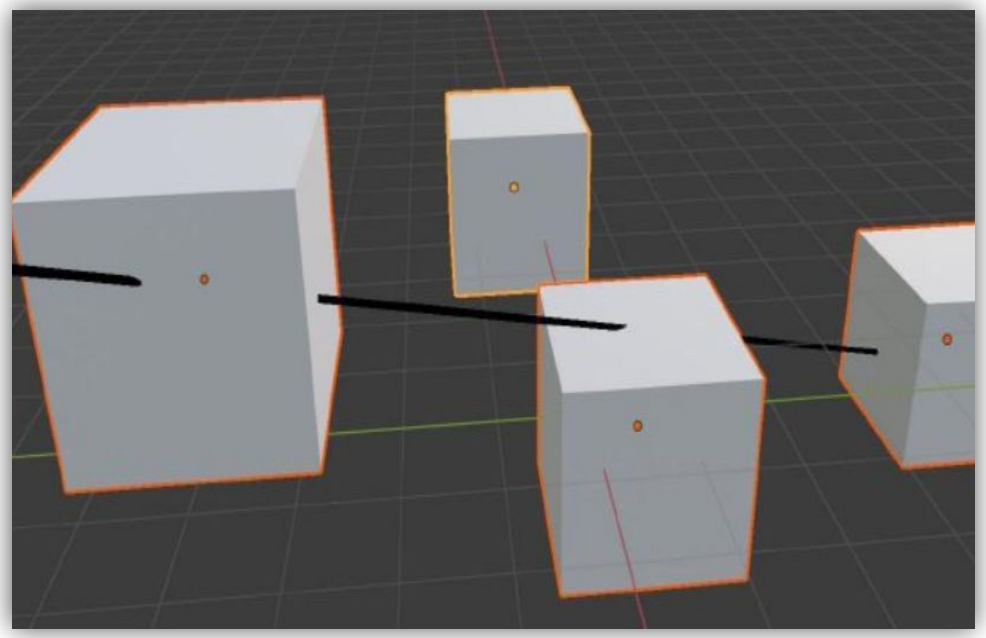
Обнаружение столкновений

Обнаружение столкновений позволяет **реагировать на столкновение двух объектов**, а также **предотвращать прохождение объектов друг через друга**.



Один из способов обнаружения столкновений – создать **вспомогательные геометрические объекты**, такие как лучи, плоскости, сферы и многоугольники, исключительно с целью выполнения тестов на столкновение. Луч особенно полезен для **выбора объектов с экрана**, так как мы можем создать луч, который начинается с точки зрения камеры и распространяется на экран, а затем определять какие объекты он пересекает.





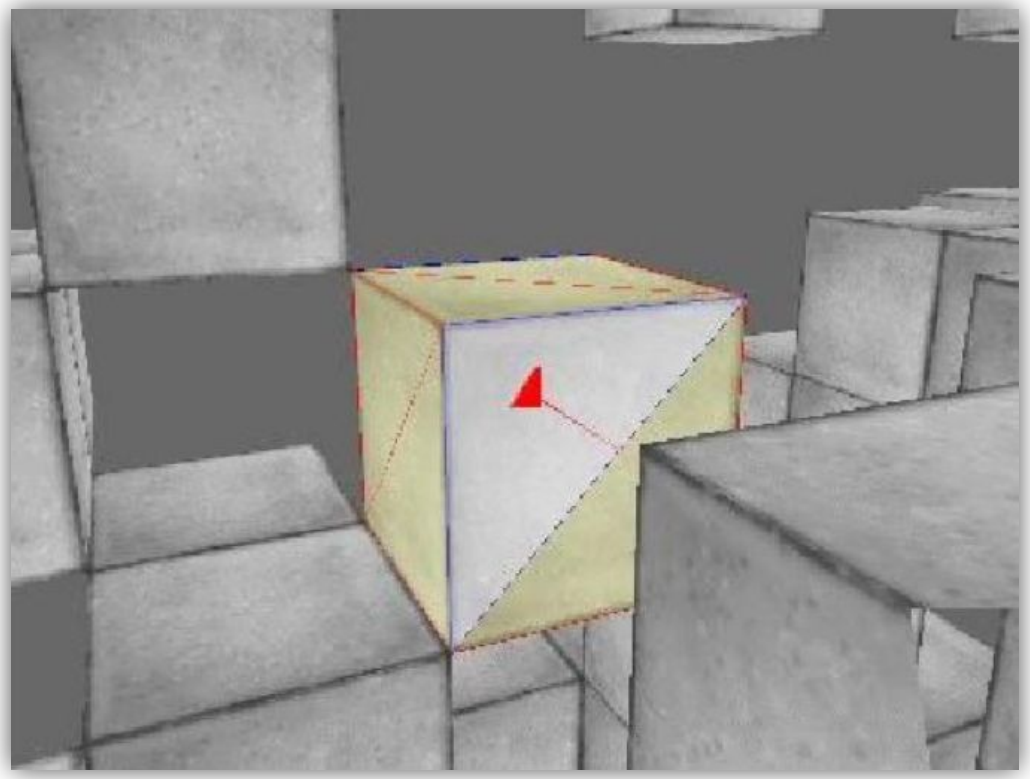
Мы решаем, как разделять мир на объекты «ОТ» и на объекты «ДО». Обычно **объекты «ОТ»** – это движущиеся объекты в сцене, а статические объекты, такие как стены – это **объекты «ДО»**.

По соображениям производительности рекомендуется минимизировать количество объектов «ОТ» в конкретной сцене. Например, сам игрок обычно является объектом «ОТ».

Пример программы, обнаруживающей столкновения:

1.Создание обходчика столкновений.
2.Создание тела столкновения – луча (объекта «ОТ»)
3. Размещение на нём камеры
4. Направление луча в центр экрана
5.Если обходчик находит столкновения, то находится узел ближайшего объекта «ДО»
6. Прокладывается прямая, перпендикулярная к касательной плоскости, к поверхности объекта «ДО».

Результат работы программы:
Красный квадрат – точка пересечения луча и блока.
Красно белая линия – вектор нормали в этой точке.



Определение столкновения с блоками

Для определения столкновений игрока с блоками можно использовать **сферу** в качестве **тела столкновения**, привязав её к центру камеры.

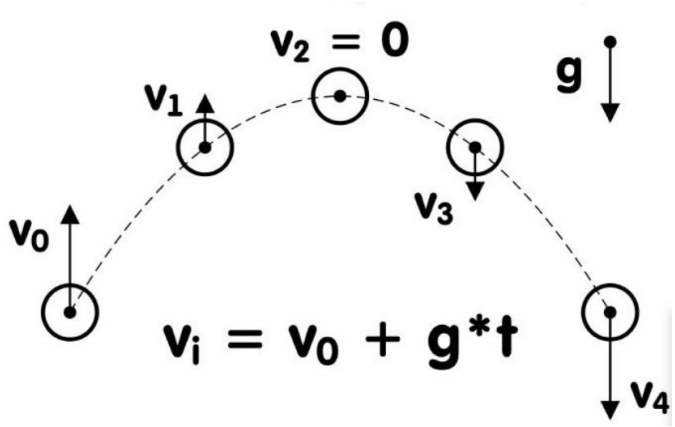
Тогда при перемещении камеры обходчик столкновений будет обнаруживать **события столкновения с блоками**, возникающие каждый раз, когда игрок будет приближаться к любому блоку на **расстояние меньшее радиуса сферы**.

Перед смещением камеры мы можем сохранить её **текущее положение**, когда она еще не сталкивалась с блоками. Тогда **сразу после смещения** камеры мы можем проверить, **произошло ли столкновение**. В этом случае необходимо вернуться на **прежнюю позицию** камеры.

Гравитация

Эффект гравитации заключается в том, что на тело действует **сила тяжести**, заставляющая его падать вниз с увеличивающейся скоростью, т.е. с **ускорением свободного падения**.

Для смещения камеры под силой гравитации необходимо вычесть из её **текущей высоты** значение **скорости падения**, а затем увеличить **скорость падения** на **ускорение падения**.



Дополнительные инструменты Panda3D

В проекте используются следующие строки:

‘window-title «имя»’ – установка заголовка окна приложения
‘sync-video false’ – отключение вертикальной синхронизации
‘show-frame-rate-meter true’ – отображение счетчика кадров в секунду
‘cursor-hidden true’ – скрытие курсора мыши
‘win-size 1000 700’ – установка размера окна приложения

Для передачи уникальных индексов блоков используются теги узлов Panda3D

setTag(ключ, значение) – записывает стоковое значение в узел по указанному ключу.
getTag(ключ) – извлекает записанное значение, соответствующее указанному ключу.