Моделирование визуальной одометрии мобильного робота в среде Blender Game Engine

Д. С. Курушин[1], Е. В. Долгова[2], Р. А. Файзрахманов[3]
Пермский национальный исследовательский политехнический университет
dan973@yandex.ru[1], elena@dolgova.info[2], itas@pstu.ru[3]

***Аннотация*. Решается задача управления мобильным роботом, который способен к автономному поведению. На практике часть отладки его системы одометрии целесообразно проводить не на реальной местности, а на виртуальном полигоне со статическими и динамическими объектами. Разработка такой подсистемы была разделена на последовательно решаемые задачи, связанные с маневрированием объекта. Далее рассматривается решение этих задач путем моделирования в среде Blender Game Engine. В результате получена модель технического зрения, создана и наполнена набором правил база знаний и получена работоспособная подсистема визуальной одометрии.**

Ключевые слова: мобильный робот; визуальная одометрия; виртуальный полигон; движок BGE; система технического зрения

Рассмотрим задачу управления мобильным роботом, который способен к автономному поведению и перемещается в постоянно меняющихся условиях окружающей среды. В этом случае исключительно важное значение имеет задача навигации, и ее решение на программном уровне существенно облегчает управление. Одометрия – это часть системы навигации, которая предполагает использование данных о движении устройства для оценки его перемещения. На практике часть отладки такой системы целесообразно проводить не на реальной местности, а на виртуальном визуализированном полигоне, который легко изменяем и позволяет эффективно моделировать виртуальные препятствия и разнообразные полезные для отладки управления ситуации.

Рассмотрим реализацию подсистемы визуальной одометрии, созданную на базе движка Blender Game Engine (BGE).

Так как дизайн системы в данном случае не моделируется, в качестве модели мобильного робота выберем простой полигональный объект (например, куб или параллелепипед). Такой объект является динамическим, то есть обладают размером, массой, скоростью, ускорением. Также при реализации виртуального полигона будем использовать статические объекты, которые выполняют роль препятствий, такие объекты характеризуются только размером. В BGE имеется несколько вариантов задания физики объектам (рис. 1):

- static – статическая физика;
- dynamic –линейная физика;
- rigid body – жесткое тело – линейная и угловая физика;
- soft body – мягкое тело;
- occluder – объект, который не будет подвергнут рендеру;
- sensor – схожесть с физикой объектов, которые лежат в основе сенсоров near и radar;
- navigation mesh – навигационная сетка;
- character – простая кинематика, подходящая для персонажей игр;
- no collision – объекты, не затронутые физикой [1].

Для моделирования системы технического зрения (СТЗ) в рамках BGE был выбран сенсор radar, который дает возможность ориентироваться в пространстве. С помощью него можно проверить, находится ли в некоторой близости другой объект. Выбранный сенсор представляет собой конус, вершина которого лежит в центре объекта, к которому привязан данный сенсор (рис. 2). Сенсор radar реагирует только на те объекты, которые являются акторами. Такая модель, конечно, является упрощенной, однако имеет преимущество, так как сенсор radar является элементом встроенного инструментария, что облегчает процесс построения подсистемы визуальной одометрии. Кроме того, функционал этого средства вполне достаточен для решения поставленной задачи и упрощение оправдано.

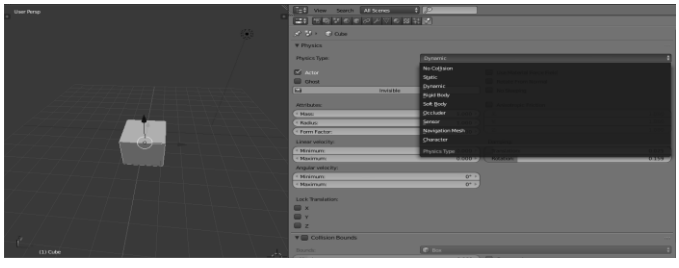

Рис. 1. Физические свойства объектов

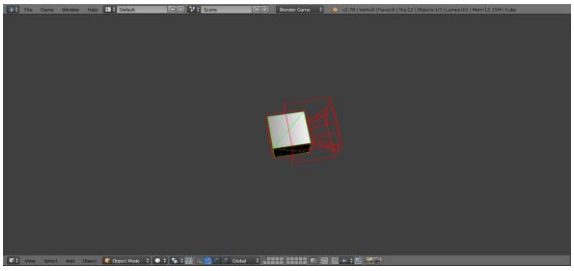

Рис. 2. Сенсор radar

Кроме того, движок BGE поддерживает следующие сенсоры:

- actuator – срабатывает, когда соответствующий актуатор активен;
- always – срабатывает «всегда» (каждый логический тик);
- collision – срабатывает при соприкосновении объектов;
- delay – срабатывает с определенной периодичностью;
- joystick – срабатывает на движение джойстика или на нажатие кнопки;
- keyboard – срабатывает на нажатие клавиш клавиатуры;
- message – срабатывает при получении подходящего сообщения;
- mouse – срабатывает на сигналы, посылаемые мышью;
- near – срабатывает на объекты, которые попадают в настраиваемую дистанцию обнаружения;
- property – срабатывает на изменения свойств объекта;
- random – срабатывает случайным образом;
- ray – срабатывает, если направленный луч встречает на пути объект, обладающий конкретным свойством или материалом [2].

Разработка подсистемы визуальной одометрии была разделена практические задачи, которые решались последовательно.

A. Задача первая

Прототип мобильного робота, падая по наклонной плоскости, должен увернуться от статического препятствия. На рис. 3 статическим препятствием является объект A, а прототипом – B.

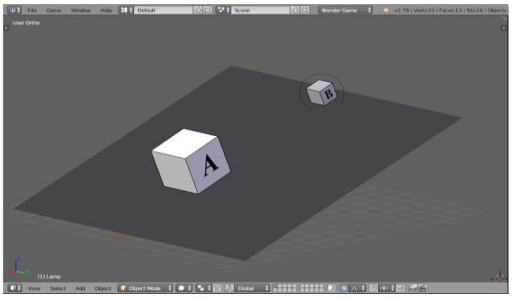

Рис. 3. Схема задачи №1

Для того чтобы управлять движением автономного мобильного робота в произвольной среде требуется база знаний, размер которой зависит от сложности решаемых задач, а характер – от их специфики. Так, например, в [3–5] рассматривается создание мобильного робота, проверка его на виртуальном полигоне и управление им на основе активной семантической сети, однако, не рассматриваются детали представления одометрии в базе знаний устройства. Чтобы восполнить этот пробел, обратимся к форме представления знаний, которые используются при решении задачи. Одним из адекватных задаче моделей знаний являются логические правила. В данном случае Они будут иметь вид:

$$близкоОбъект(X) <= (X < 2)$$
$$(увернутьсяМРК[X] == сила) <= близкоОбъект(X)$$
$$(\text{увернутьсяМРК[X]} == 0) <= \sim(\text{близкоОбъект(X)})$$

Причем, сенсор delay используется, чтобы с определенной частотой обращаться к контроллеру BGE. Контроллером в этой ситуации является скрипт на языке Python, который управляет роботом. Близость его к препятствию определяется по координатам. Такой подход моделирует получение координат объектов с помощью спутниковой системы навигации.

B. Задача вторая

Два прототипа робота, находящиеся, например, на парковке, должны покинуть парковочное место, не задев впередистоящий автомобиль и уступив дорогу проезжающему автомобилю. На рис. 4 прототипами робота являются объекты A, периодически проезжающим автомобилем – объект B, стоящими на парковке автомобилями – объекты C.

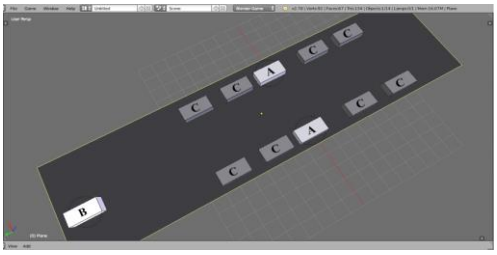

Рис. 4. Схема задачи №2

На рис. 5 показано, что один робот на виртуальном полигоне уже покинул свое парковочное место и двигается по проезжей части, а второй стоит и пропускает проезжающую машину.

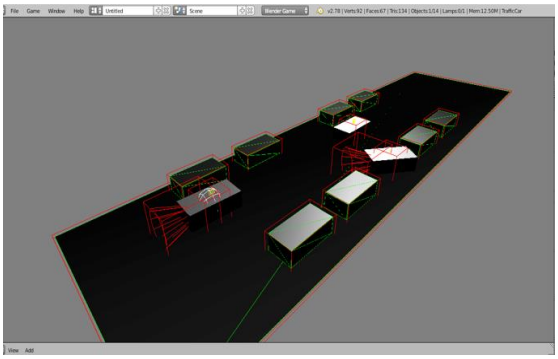

Рис. 5. Физика задачи №2

Правила для базы знаний имеют следующий вид:

радарСработал(X) <= *(X == True)*
(повернутьМРК[X,МРК] == 0.03) <=
радарСработал(X) & (МРК == "Robot2")
(повернутьМРК[X,МРК] == -0.03) <=
радарСработал(X) & (МРК == "Robot1")
(повернутьМРК[X,МРК] == 0.00) <= ~ (радарСработал(X))

В качестве сенсоров здесь использовались: сенсор delay, сенсор keyboard, сенсор property, сенсор radar. Сенсор delay отвечает за периодическое обращение к контроллеру, который проверяет достижение проезжающим автомобилем конца дороги. Сенсор keyboard отвечает за начало движения робота по команде оператора виртуального полигона. Сенсор property отвечает за выполнение некоторых действий вследствие срабатывания определенного свойства. Сенсор radar отвечает за обнаружение впередистоящего автомобиля.

Контроллерами в этой ситуации являются скрипты на языке Python и связки and, которые связывают сенсор и актуатор. Актуатор перемещает МРК после получения сигнала от контроллера, которого в свою очередь вызвал сработавший на некоторое условие сенсор.

Близость МРК к проезжающему автомобилю высчитывалась по имеющимся координатам. Такой подход моделирует получение координат объектов с помощью спутниковой системы навигации. Определение препятствия в виде впередистоящего на парковке автомобиля происходило с помощью сенсора radar, который моделирует простейшую СТЗ. Таким образом была показана возможность визуальной навигации в дополнение к существующим.

C. Задача третья

Прототип робота должен достичь конца извилистой дороги, при этом не столкнувшись со статическими препятствиями на своем пути. На рис. 6 прототипом такого робота является объект A, статическими препятствиями – объекты B, началом дороги – C, концом дороги – D.

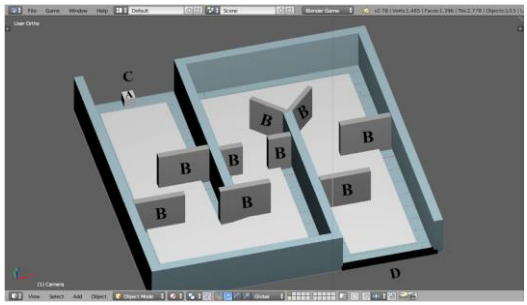

Рис. 6. Схема задачи №3

На рис. 7 видно, как робот начинает свое движение по дороге. Также на рисунке можно видеть функционирующие сенсоры radar.

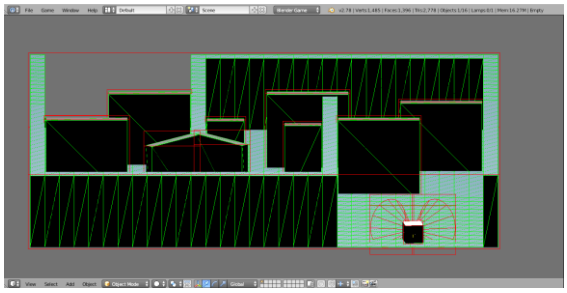

Рис. 7. Физика задачи №3

Правила в данной задаче аналогичны правилам, которые были приведены в примерах предыдущих задач. В качестве сенсоров в данной задаче использовались: сенсор keyboard, сенсор property, сенсор radar. Сенсор keyboard отвечает за начало движения робота по команде оператора виртуального полигона. Сенсор property отвечает за выполнение некоторых действий вследствие срабатывания определенного свойства. Сенсор radar отвечает за обнаружение препятствия.

Контроллерами в этой ситуации являются скрипты на языке Python и связки and, которые связывают сенсор и актуатор. Актуатор перемещает робота после получения сигнала от контроллера, которого в свою очередь вызвал сработавший на некоторое условие сенсор.

Таким образом, используя инструментарий движка BGE, удалось:

- смоделировать его систему технического зрения в упрощенном виде, с помощью встроенных средств;
- создать базу знаний и наполнить ее начальным набором правил;
- получить функционирующую подсистему визуальной одометрии прототипа робота как совокупность СТЗ и экспертной системы (ЭС) с помощью библиотеки pyDatalog, вспомогательных инструментов Blender Game Engine и связующего кода на языке программирования Python 3.

В дальнейшем возможно совершенствование и расширение ЭС, детализация визуальной модели робота и улучшение модели СТЗ, а также использование полученных результатов при построении траектории движения с учетом поверхности местности.

Список литературы

[1] Physics – Blender Manual // URL: https://docs.blender.org/manual/en/dev/physics/index.html (дата обращения: 14.06.2017).

[2] Doc:RU/2.6/Manual/Game Engine/Logic/Sensors // BlenderWiki URL: https://wiki.blender.org/index.php/Doc:RU/2.6/Manual/Game_Engine/Logic/Sensors (дата обращения: 14.06.2017).

[3] Dolgova E.V., Fayzrakhmanov R.A., Kurushin D.S. Decision-making in autonomous mobile robot control system based on active semantic network (Принятие решений в системе управления автономным мобильным роботом на основе активной семантической сети)/ Proceedings of 2017 20th IEEE International Conference on Soft Computing and Measurements, SCM 2017

[4] Курушин Д.С., Кондаков Д.В., Долгова Е.В. Разработка аппаратной части автономного мобильного роботизированного комплекса/ Информационно-измерительные и управляющие системы. 2015. Т. 13. № 9. С. 45-50.

[5] Головкова Е.В., Долгова Е.В., Злобин С.А., Кальсин А.О., Курушин Д.С. Экспериментальное исследование метоодв построения пути для многофункциональной интеллектуальной самоходной мобильной платформы/ В мире научных открытий. 2015. № 10.2 (70). С. 663-678.