Министерство образования Тульской области

Муниципальное бюджетное учреждение дополнительного образования «Дворец детского (юношеского) творчества» города Новомосковска

Тульской области

Муниципальное бюджетное общеобразовательное учреждение «Лицей»

научно-исследовательская работа

«Симулятор управления марсоходом Curiosity»

**Автор**:

Кривошеин Александр Дмитриевич

учащийся 10 б класса МБОУ «Лицей»,

МБУ ДО «ДДЮТ» г. Новомосковска Тульской области

**Руководитель**:

Николаева Наталья Викторовна

педагог дополнительного образования МБУ ДО «ДДЮТ»

г. Новомосковска Тульской области

Новомосковск, 2016 г.

Оглавление

[Аннотация 3](#_Toc449535046)

[1. Введение 4](#_Toc449535047)

[3. Практическая часть 7](#_Toc449535048)

[4. Заключение 12](#_Toc449535049)

[5. Список литературы 13](#_Toc449535050)

[Приложения 14](#_Toc449535051)

# Аннотация

Цель проекта: создать компьютерный симулятор управления марсоходом Curiosity. Программа позволяет пользователю тренироваться в дистанционном управлении марсоходом. Программа доступна для пользователей с любым уровнем подготовки. Управление происходит с помощью клавиатуры или джойстика.

Работа разделена на несколько этапов:

1. Разработана идея и общий вид симулятора.
2. Изучен путь движения по Марсу марсоходом Curiosity и выбрана местность, по которой осуществляется движение в симуляторе.
3. Создана 3D модель выбранной поверхности Марса.
4. Создана 3D модель Curiosity.
5. Запрограммирована модель марсохода максимально приближенная в управлении к настоящему Curiosity.
6. Разработана инструкция по управлению марсоходом.
7. Программа Адаптирована к управлению с помощью джойстика.
8. Скомпилирована программа для операционной системы Windows.

Таким образом, создан симулятор управления марсоходом Curiosity. Его доступность позволяет всем пользователям попробовать управлять марсоходом. Это позволяет внедрять космические технологии в общество и пропагандировать космонавтику, как развивающуюся отрасль.

В работе использован 1 рисунок. В приложении представлены 10 рисунков и 3 скрипта на языке С#.

Источники информации:

1. Кемурджиана, А. Л. ”Планетоходы” –Москва: «Машиностроение», 1993 – 397 с.
2. URL: http://mars.nasa.gov/msl/

# Введение

Человечество уже в прошлом веке начало освоение космоса. Прошло 55 лет с первого полёта человека в космос, и в наше время технологии идут широким шагом. Идея использования внеземного транспорта принадлежит ученым и [конструкторам](http://unusauto.ru/lunohody-marsohody.htm), реализовавшим космическую программу СССР. "Программа исследования Луны", озвученная в1958 году – С. П. Королевым, предусматривала ряд изысканий и экспериментов, которые давали возможность более подробно изучить природу этого небесного тела. Планетоходы, благодаря своей мобильности, позволяли расширить зону исследуемой поверхности спутника Земли.

Изготовителем первого планетохода стало знаменитое НПО

им. С. А. Лавочкина – преемник легендарного ОКБ-1 в сфере межпланетных исследований. Конструкторское бюро этого научно-производственного объединения аккумулировало все теоретические и практические разработки советских НИИ. Аппарат управлялся удаленно, специально отобранным экипажем, который размещался в наземном центре. Общая численность экипажа (с учетом дублирующих пилотов) составила 11 человек. Для их обучения создавались специальные программы и тренажеры.

Цель работы: создать программу-симулятор для обучения управлению марсоходом.

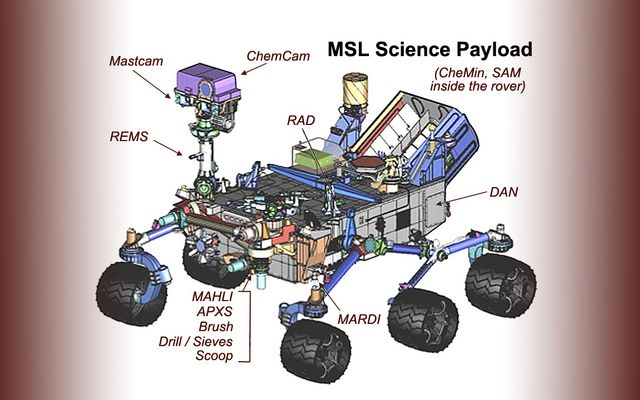
Задачи работы:

* выбрать марсоход для создания программы;
* изучить рельеф марсианской поверхности;
* изучить управление планетоходами с Земли;
* создать трёхмерную модель поверхности марса на графическом движке Unity5;
* создать трёхмерную модель марсохода;
* скомпилировать программу;
* создать сайт и разместить на нём программу.

1. **Теоретическая часть**

В программе управление будет предоставлено на примере марсохода Curiosity. Это проект передвижной исследовательской лаборатории американского аэрокосмического агентства «NASA». Он совершил посадку в пункт своего назначения в 2012 году.

Чтобы построить наиболее правдоподобную модель поверхности Марса, я изучал космические снимки. Это позволило изучить структуру грунта и перепад высот на пути Curiosity (рис. 1 приложение)

Для создания симулятора я взял манипулятор, с помощью которого управление марсоходом станет правдоподобным. Сигнал от Земли до Марса и обратно идёт долго для человеческого восприятия, поэтому водители марсоходов ориентируются на задержку в 2-3 минуты.

На сайте NASA марсоход Curiosity представлен следующим образом:

* Камеры:
  + Мачтовая камера;
  + Ручная геологическая линзовая камера;
  + Цветная видеокамера.
* Спектрометры:
  + Альфа-спектрометр;
  + Лазерный анализатор химического состава;
  + Химическо-минеральный рентген.
* Радиационные детекторы:
  + Детектор оценки радиации;
  + Динамическое альбедо нейтронов.
* Датчики окружающей среды:
  + Станция мониторинга окружающей среды.
* Датчики атмосферы:
  + Устройство записи спуска и прибытия марсохода.

# Практическая часть

Для написания симулятора использовался графический движок Unity3d.

Для создания правдоподобной марсианской поверхности был применён встроенный в среду разработки ландшафтный дизайнер. Для придания большей правдоподобности рельефу планеты на полигональной сетке были выполнены перепады высот, в программу добавлены трёхмерные модели камней. Для текстурирования поверхности Марса использовались 4 бесшовные текстуры камня и песка.

Для создания правдоподобного неба использовался скайбокс однотонного неба. Для моделирования рабочей трёхмерной модели марсохода использован компьютерный чертёж Curiosity. Для перевода в стандартный трёхмерный формат OBJ воспользовался программным пакетом Autodesk Maya. Полигональная сетка, получившаяся после экспорта из чертежа, получилась очень детальной, требовала больших ресурсов процессора и оперативной памяти, из-за чего не могла использоваться в симуляторе. В связи с этим все детали марсохода прошли ретопологию в программе TopoGun(рис. 2 приложение), а колёса марсохода - в программном пакете Autodesk 3ds max. Это требовалось, чтобы колёса были полностью округлыми. Их поперечное сечение должно быть идеальной окружностью, для достижения этой цели стандартный цилиндрический примитив подвергался редактированию.

Для всех моделей, получившихся после ретопологии, выполнена развёртка для текстурных карт в программе UVLayout.(рис. 3 приложение) Все модели с развёртками экспортированы в 3ds max, где были собраны из отдельных деталей в готовую модель.(рис. 4 приложение) При помощи xNormal для всех частей модели созданы normal map, ambient occlusion map (рис 5. 6 приложение). Normal карта имитирует неровность поверхности модели, не используя сложную полигональную сетку. Для этого большая полигональная сетка проецируется на маленькую и записывается в текстурную карту последней. Кроме того, требовалось создать diffuse, которая отражает цвет модели. Для большей правдоподобности нам понадобится ambient occlusion map, созданная ранее в xNormal. Для этого используется программный пакет SubstancePainter. (рис. 7 приложение)

Создание анимация движения и поворота колёс в 3dsmax:

Путём создания ключевых кадров с разными показателями поворота и положения в пространстве, я получил кривую анимации. Так как получившиеся кривые не всегда корректны, они были исправлены вручную (рис. 8 приложение). Различные анимации выполнены на отдельных копиях модели, и каждая экспортирована в формат FBX.

Все получившиеся модели и текстуры экспортированы в среду разработки Unity5. Создан Animator Controller (контроллер анимации) для анимационных движений модели Curiosity. Он позволяет плавно проигрывать анимации, управляя ими из C# скрипта.

Скрипты для симулятора написаны на языке программирования C#:

1. Скрипт 1 (приложение) для перемещения марсохода в пространстве и его поворотов. Он отслеживает нажатие клавиш. В случае, если клавиши нажаты, то скрипт начинает изменение локальных координат объекта. Он контролирует перемещение курсора и поворачивает камеру.
2. Скрипт 2 отвечает за создание паузы при нажатии клавиши. Он останавливает все перемещения и выключает предыдущий скрипт, а также включает дополнительную камеру, которая показывает маленькое меню с кнопками продолжения и выхода.

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | **using UnityEngine**;  **using System.Collections**;  var paused : boolean = **false**;  var control : boolean = **true**;  var camera1 : Camera;  var camera2 : Camera;  **public class Camera\_Control** : MonoBehaviour  {  **void** Update ()  {  **if** (Input.GetKeyDown(KeyCode.Escape))  {  control = **true**;  **if** ((!paused) && (control == **true**))  {  Time.timeScale = 0;  paused = **true**;  audio.Pause();  Screen.showCursor = **true**;  control = **false**;  camera1.enabled = **true**;  camera2.enabled = **false**;  }  **if** ((paused == **true**) && (control == **true**))  {  Time.timeScale = 1;  paused = **false**;  audio.Play();  Screen.showCursor = **false**;  control = **false**;  camera1.enabled = **false**;  camera2.enabled = **true**;  }  }  }  } | |  |

3) Скрипт 3 (приложение) используется в первой сцене программы. Он отслеживает нажатие кнопок в меню и делает действия:

* Включает симулятор

Переводит пользователя на вторую сцену, в которой ему доступен в управление марсоход.

* Показывает инструкцию

Переводит пользователя на страницу инструкции, в которой ему показывает подробно описанные правила пользования программой.

* Вызывает меню настроек

Переключает камеру в сцене на меню настроек,в котором можно будет отрегулировать качество моделей, текстур, теней.

* Выключает программу

4) Скрипт 4 контролирует проигрывание анимаций у модели марсохода.

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46 | **using UnityEngine**;  **using System.Collections**;  **public class PlayerController** : MonoBehaviour {  **public** Animator anim;    **void** Update ()  {  **if**(Input.GetKeyDown (KeyCode.W))  {  anim.SetBool (“forward”, **true**);  }  **if** (Input.GetKeyUp (KeyCode.W))  {  anim.SetBool (“forward”, **false**);  }  *//------------------------------------------//*  **if** (Input.GetKeyDown (KeyCode.D))  {  anim.SetBool (“right”, **true**);  }  **if** (Input.GetKeyUp (KeyCode.D))  {  anim.SetBool (“right”, **false**);  }  *//------------------------------------------//*  **if** (Input.GetKeyDown (KeyCode.A))  {  anim.SetBool (“left”, **true**);  }  **if** (Input.GetKeyUp (KeyCode.A))  {  anim.SetBool (“left”, **false**);  }  *//------------------------------------------//*  **if** (Input.GetKeyDown (KeyCode.S))  {  anim.SetBool (“back”, **true**);  }  **if** (Input.GetKeyUp (KeyCode.S))  {  anim.SetBool («back», **false**);  }  }  } | |
|  | |  | |

После создания программы я сделал сайт, используя сервис с бесплатным доменом www.wix.com. Он удобен тем, что не редактирует напрямую HTML код. Сайт доступен по следующей ссылке URL: <http://krivosheinsasha.wix.com/curiosity-simulator>.

# Заключение

Таким образом, в работе создана программа-симулятор для обучения управлением марсохода в условиях, имитирующих реальный рельеф планеты (рис. 9, 10) Для создания программы были переведены компьютерные чертежи марсохода в трёхмерную модель. Оптимизировано количество полигонов на модели путём проведения ретопологии. Модель собрана в программном пакете 3ds max. Созданы текстурные карты, имитирующие цвет, тени и металлический отблеск деталей марсохода. Создана модель поверхности Марса. Написаны скрипты для имитирования движения Curiosity.

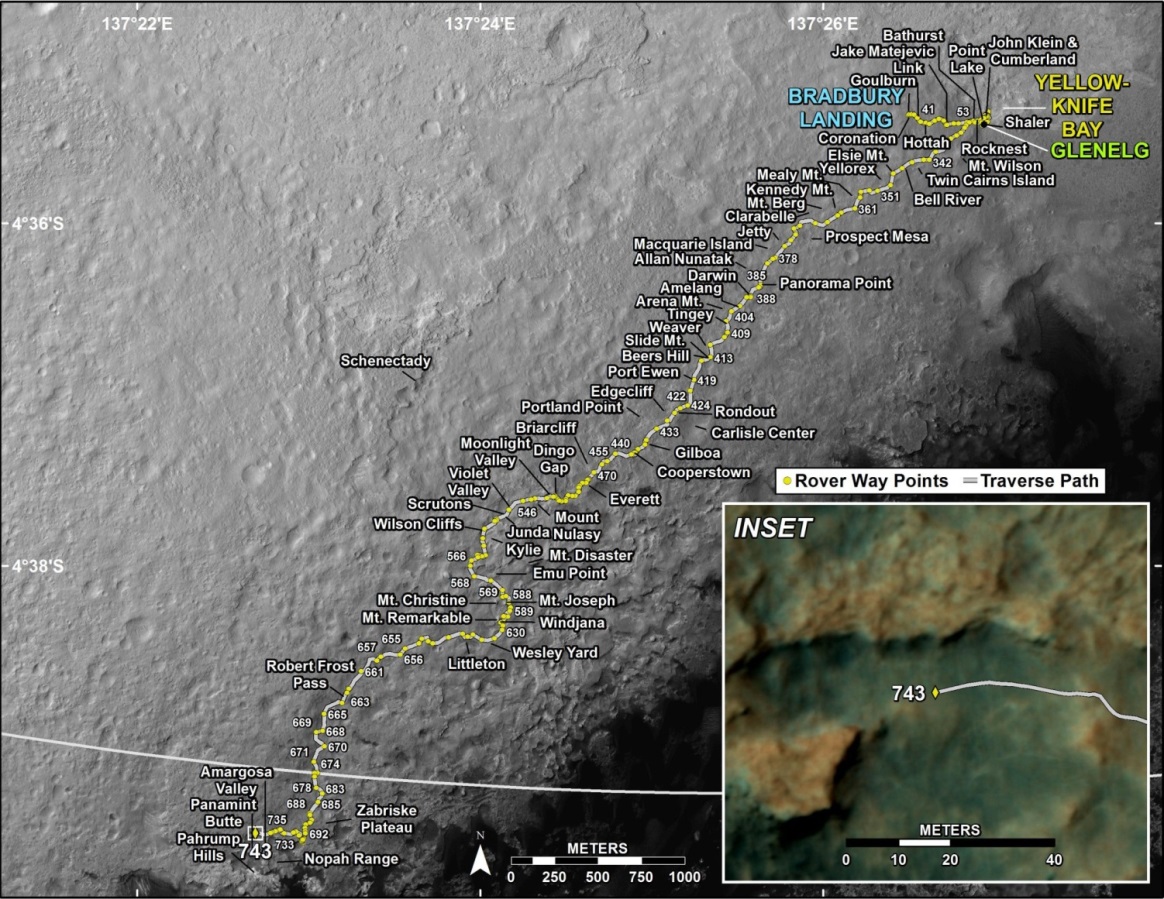
Возможность использования программы предоставлена на сайте доступном по ссылке URL: <http://krivosheinsasha.wix.com/curiosity-simulator>.

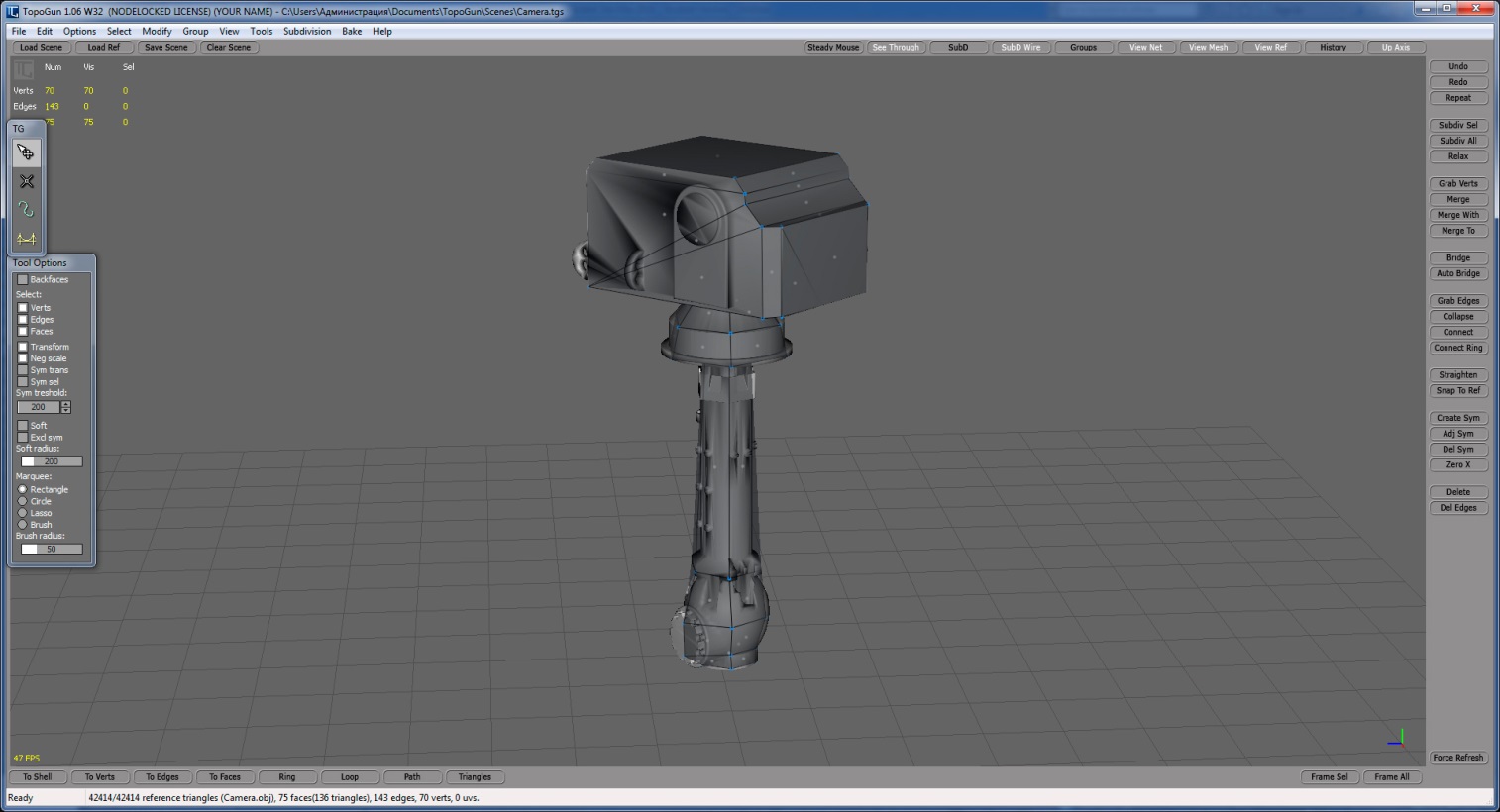
# Список литературы

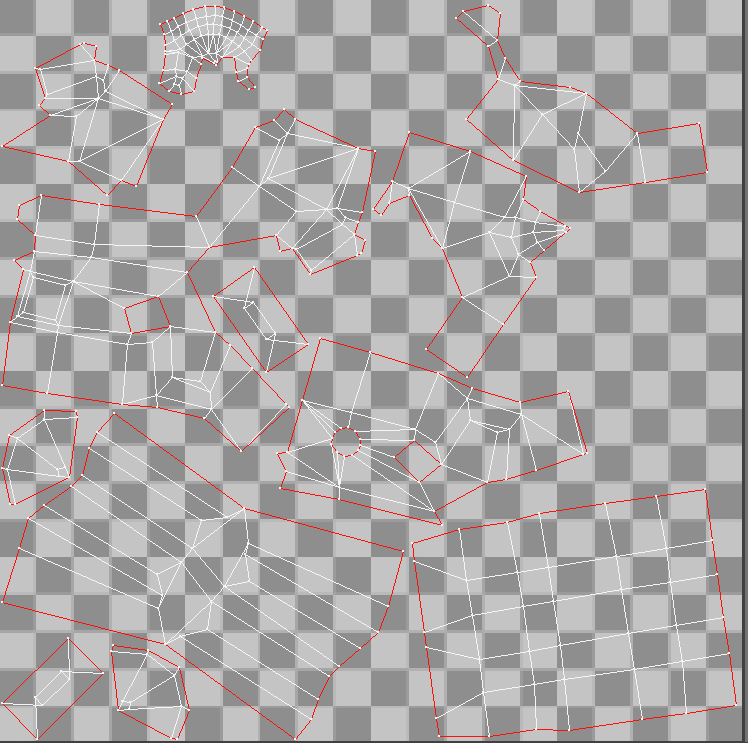
1. Кемурджиана, А. Л. «Планетоходы» –Москва: «Машиностроение», 1993 – 397 с.
2. URL: http://mars.nasa.gov/msl/

# Приложения

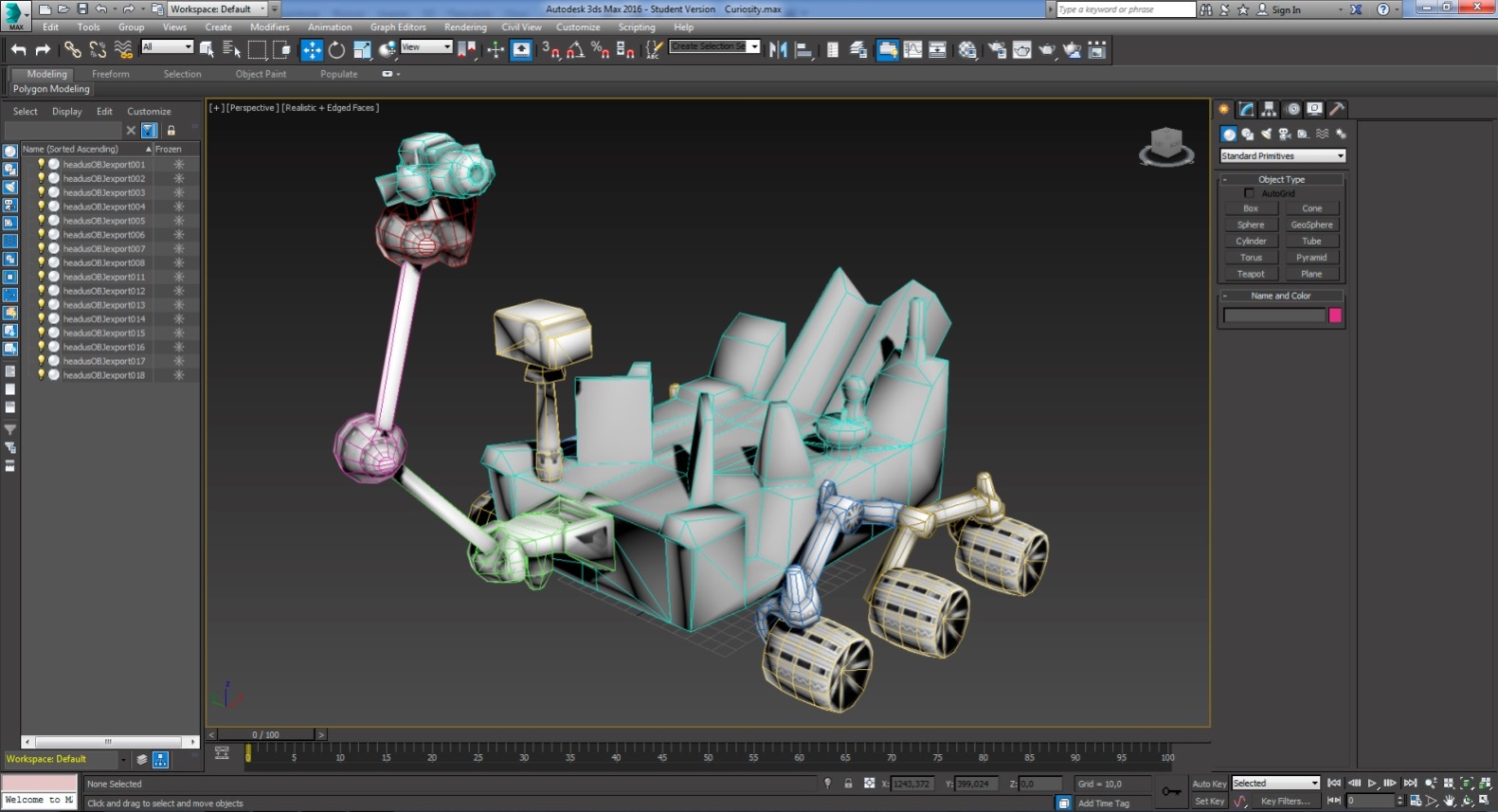
**Рис. 1. Путь движения Curiosity**

****

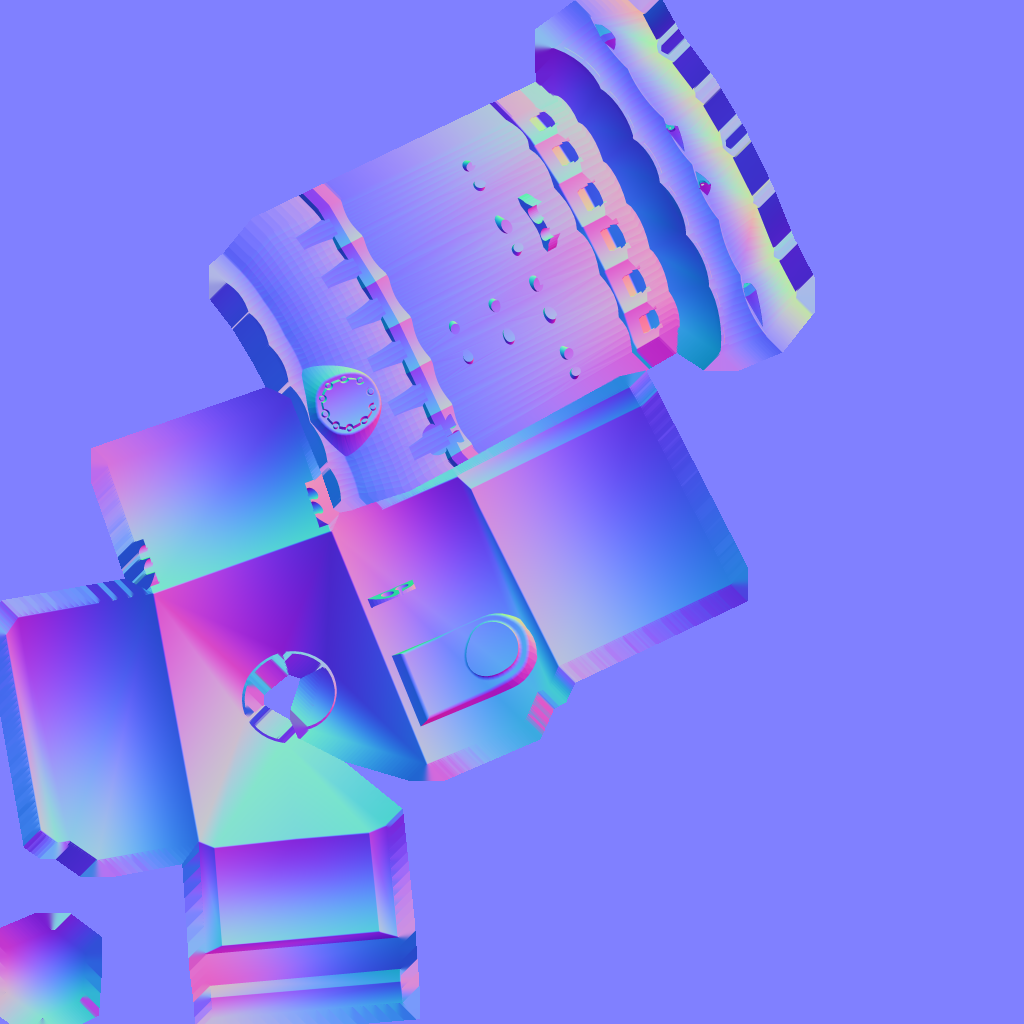
**Рис. 2. Ретопология камеры марсохода**

**Рис. 3. Развёртка детали**

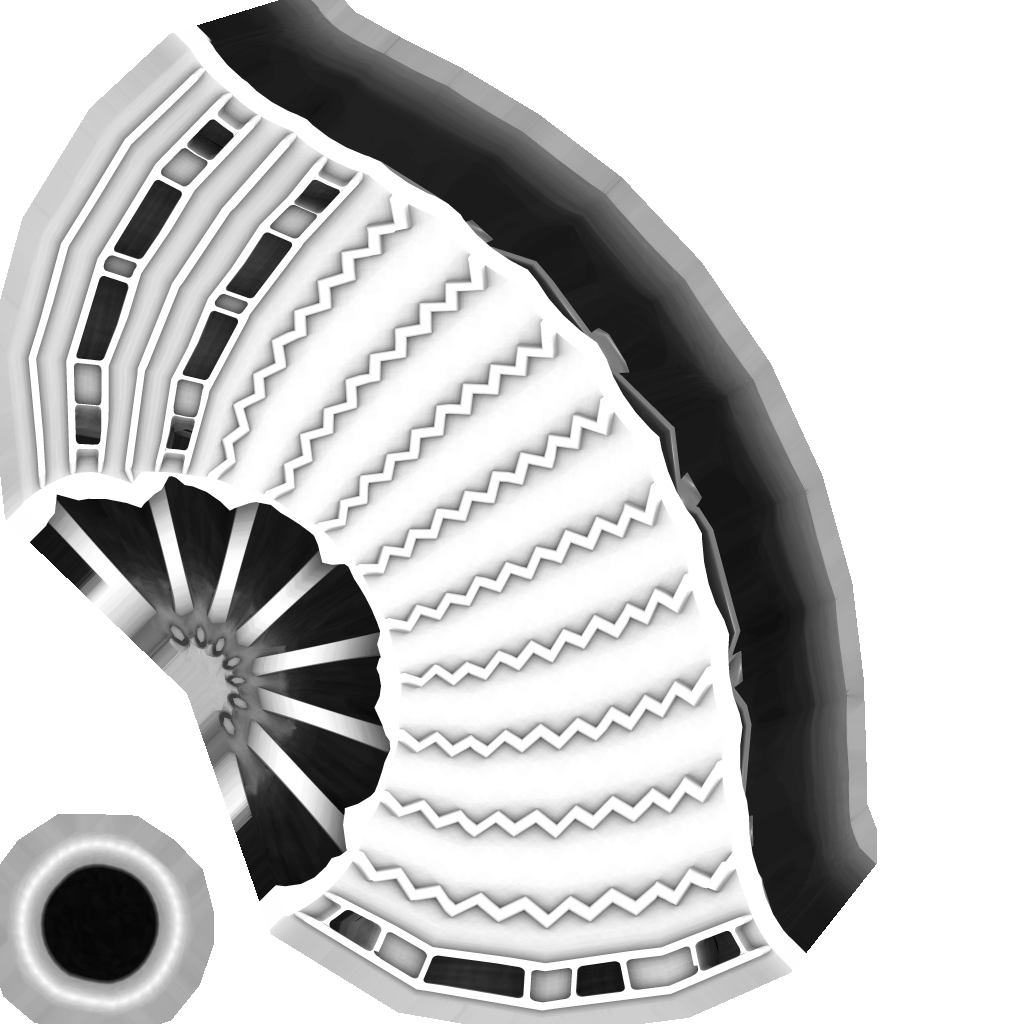
**Рис. 4. Модель собранная в 3dsmax**

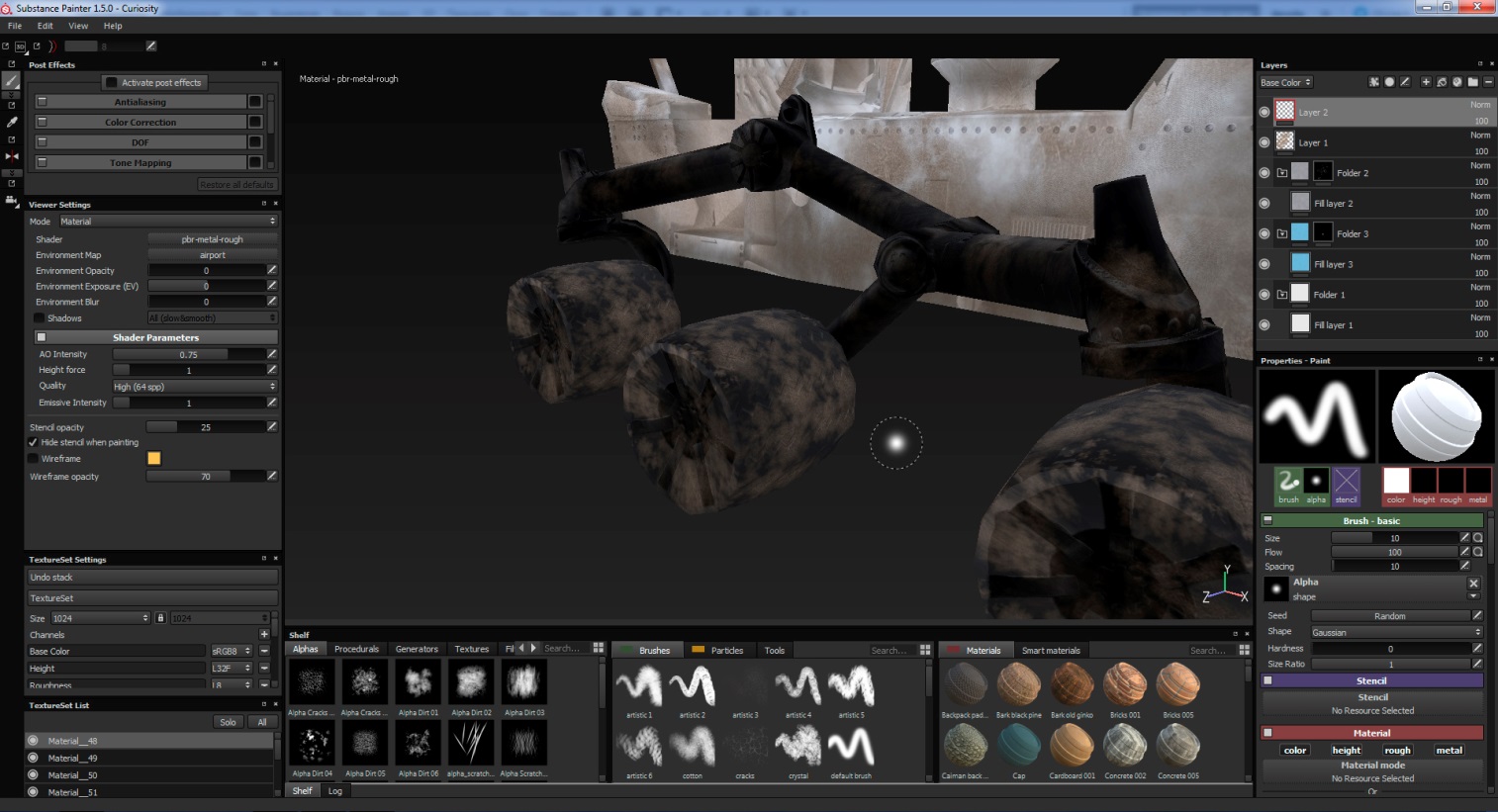
****

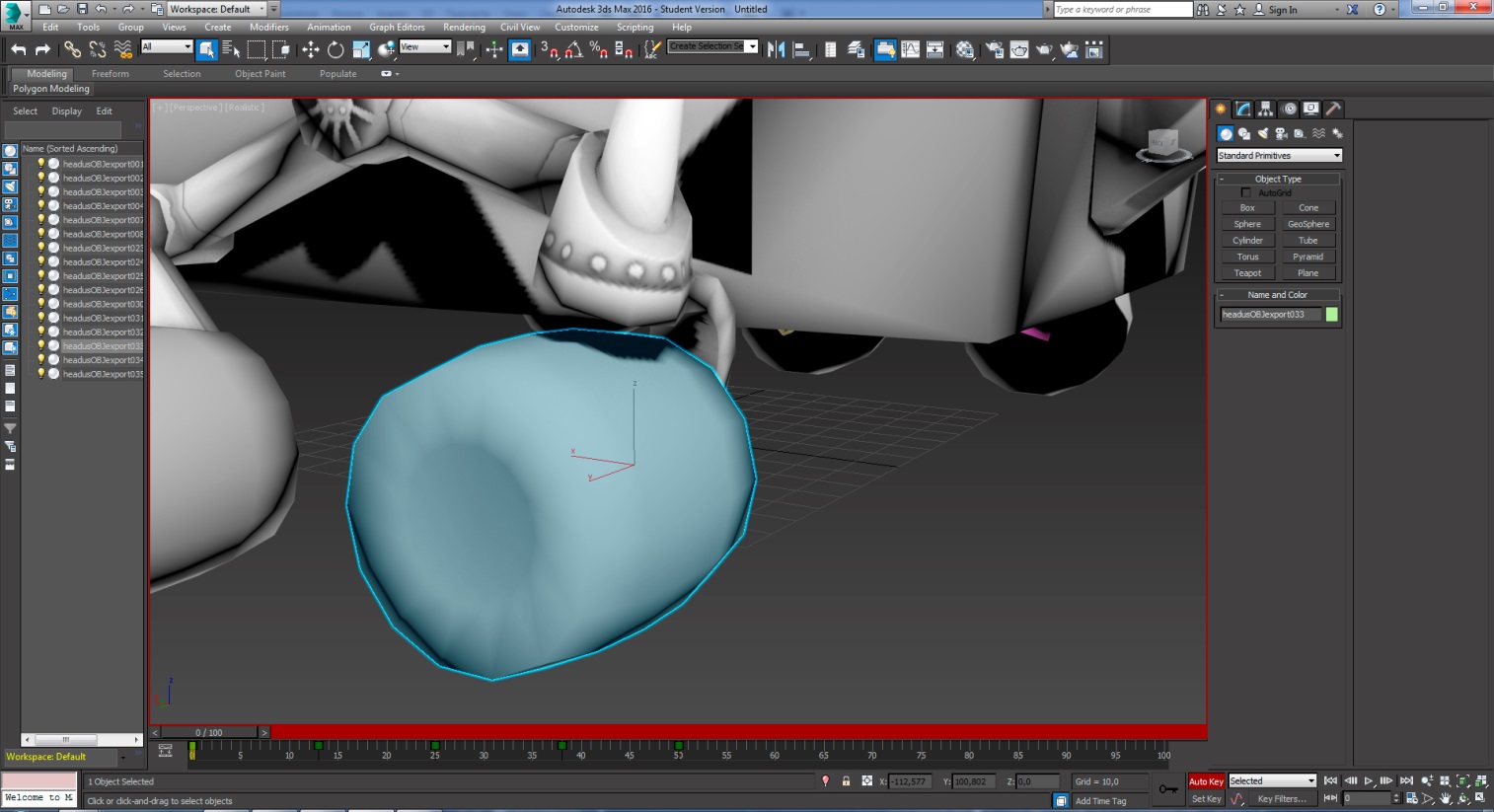
**Рис. 5. Normal map**

****

**Рис. 6. Ambient occlusion map**

****

**Рис. 7. Создание diffusialmap**

**Рис. 8. Процесс создания анимации**

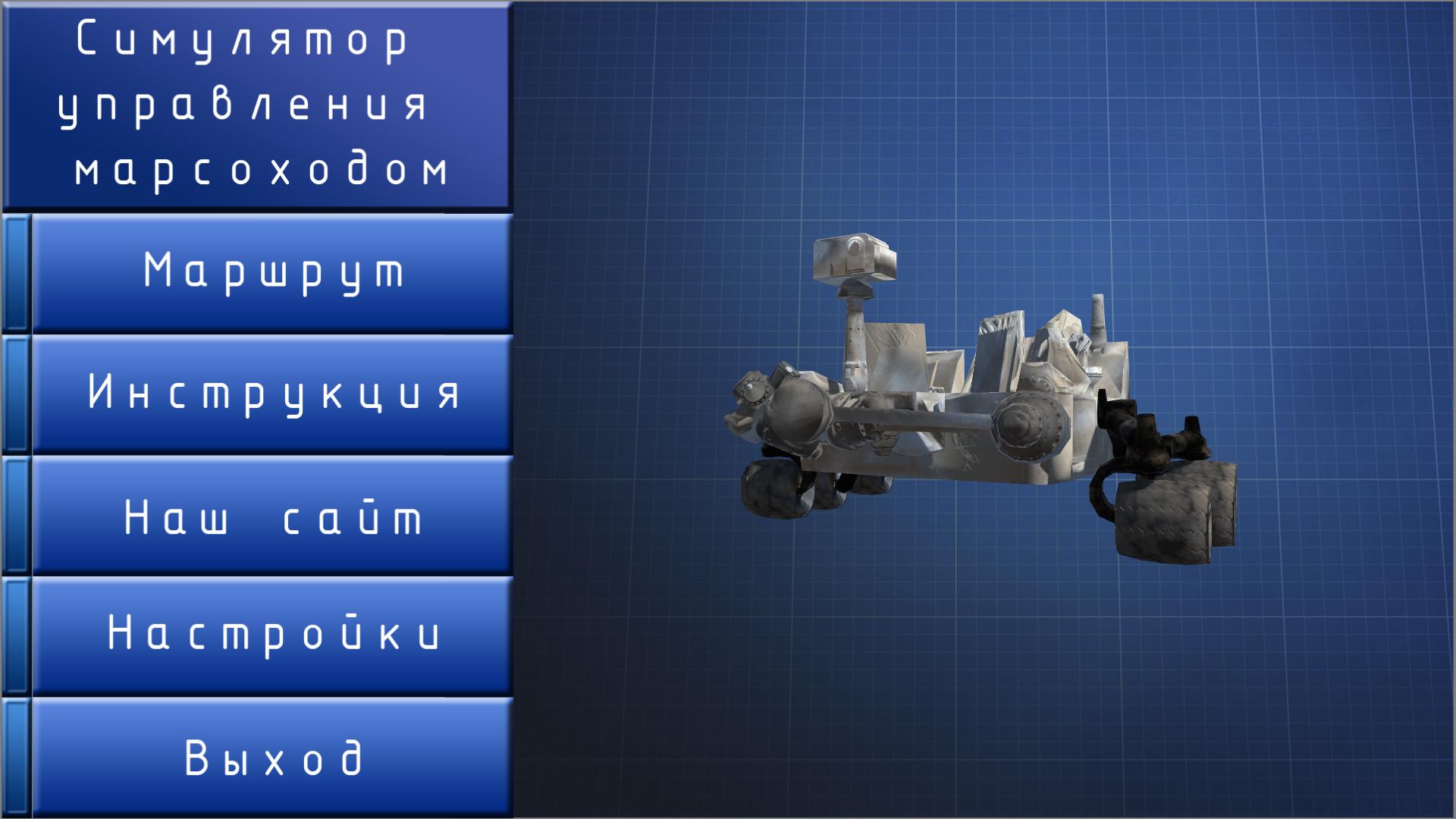
**Скрипт 1**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58 | **using UnityEngine**;  **using System.Collections**;  **public class TransPlayer** : MonoBehaviour {  **public** bool right;  **public** bool left;  **public** float forward\_speed;  **public** float back\_speed;  **public** float rotation\_speed;  **void** Start()  {  right = **false**;  left = **false**;  }  **void**Update ()  {  **if** (Input.GetKey(KeyCode.W))  {  transform.Translate(forward\_speed \* Time.deltaTime, 0f, 0f);  **if** (left == **true**)  {  transform.Rotate(0, -rotation\_speed \* Time.deltaTime, 0);  }  **if** (right == **true**)  {  transform.Rotate(0, rotation\_speed \* Time.deltaTime, 0);  }  }  **if** (Input.GetKey(KeyCode.S))  {  transform.Translate(-back\_speed \* Time.deltaTime, 0f, 0f);  **if** (left == **true**)  {  transform.Rotate(0, rotation\_speed \* Time.deltaTime, 0);  }  **if** (right == **true**)  {  transform.Rotate(0, -rotation\_speed \* Time.deltaTime, 0);  }  }  **if** (Input.GetKey(KeyCode.A))  {  left = **true**;  }  **if** (Input.GetKeyUp(KeyCode.A))  {  left = **false**;  }  **if** (Input.GetKeyDown(KeyCode.D))  {  right = **true**;  }  **if** (Input.GetKeyUp(KeyCode.D))  {  right = **false**;  }  }  } |

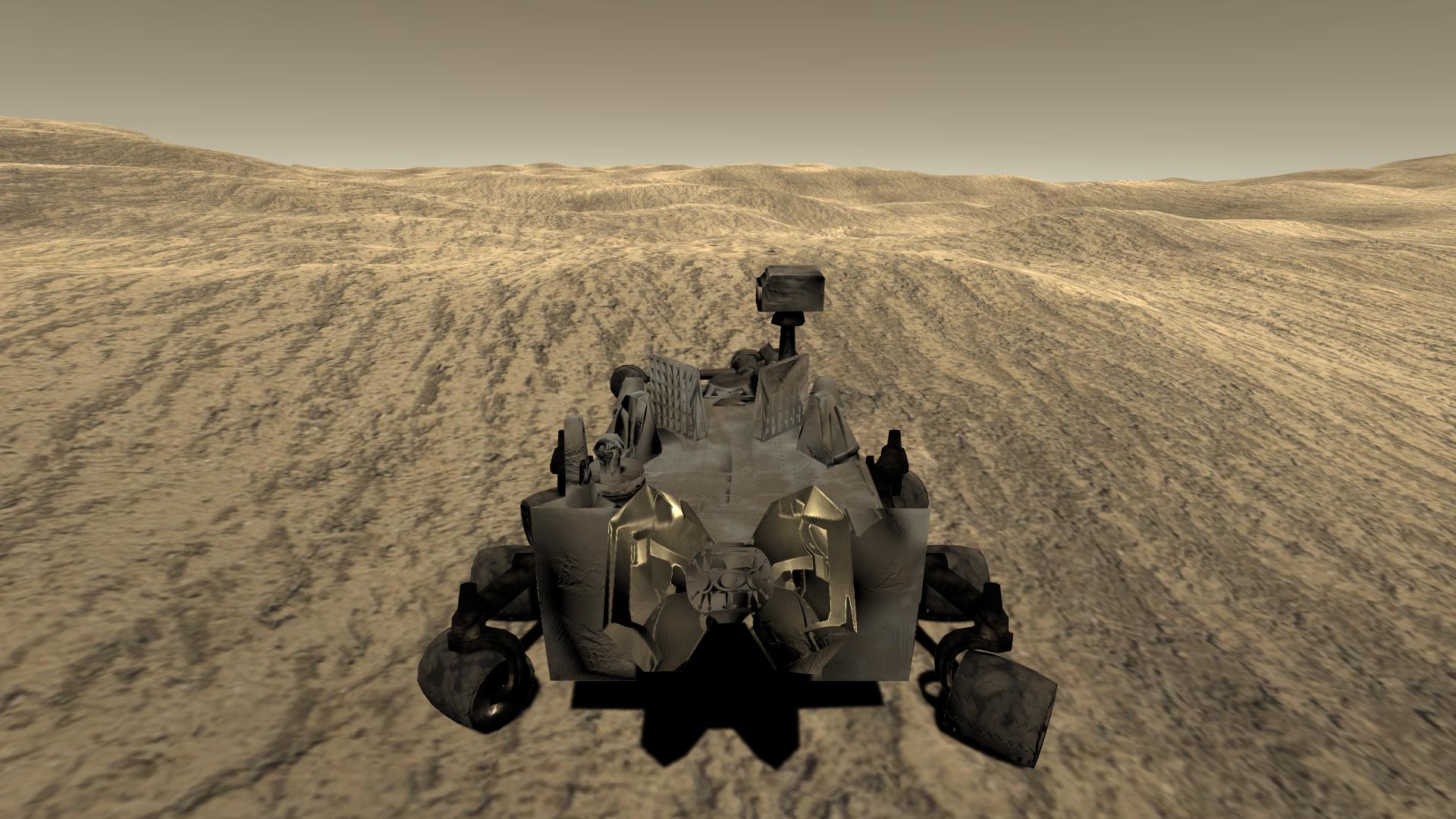
**Скрипт 3**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62 | **using UnityEngine**;  **using System.Collections**;  **public class Control** : MonoBehaviour  {  **public** GameObject looker;  **public** GameObject button1;  **public** GameObject button2;  **public** GameObject button3;  **public** GameObject button4;  **public** GameObject button5;  **void** Start()  {  Screen.SetResolution(1920, 1080, **true**);  }  **void** exit()  {  Application.Quit();  }  **void** play()  {  Application.LoadLevel(1);  }  **void** option\_true()  {  UIPanel other = looker.GetComponent<UIPanel>();  UIButton other1 = button1.GetComponent<UIButton>();  UIButton other2 = button2.GetComponent<UIButton>();  UIButton other3 = button3.GetComponent<UIButton>();  UIButton other4 = button4.GetComponent<UIButton>();  UIButtonMessage other14 = button4.GetComponent<UIButtonMessage>();  UIButton other5 = button5.GetComponent<UIButton>();  UIButtonMessage other15 = button4.GetComponent<UIButtonMessage>();  other.enabled = **true**;  other1.enabled = **false**;  other2.enabled = **false**;  other3.enabled = **false**;  other4.enabled = **false**;  other14.enabled = **false**;  other5.enabled = **false**;  other15.enabled = **false**;  }  **void** option\_false()  {  UIPanel other = looker.GetComponent<UIPanel>();  UIButton other1 = button1.GetComponent<UIButton>();  UIButton other2 = button2.GetComponent<UIButton>();  UIButton other3 = button3.GetComponent<UIButton>();  UIButton other4 = button4.GetComponent<UIButton>();  UIButtonMessage other14 = button4.GetComponent<UIButtonMessage>();  UIButton other5 = button5.GetComponent<UIButton>();  UIButtonMessage other15 = button4.GetComponent<UIButtonMessage>();  other.enabled = **false**;  other1.enabled = **true**;  other2.enabled = **true**;  other3.enabled = **true**;  other4.enabled = **true**;  other14.enabled = **true**;  other5.enabled = **true**;  other15.enabled = **true**;  }  } |

**Рис. 9. Меню симулятора**

****

**Рис. 10. Симулятор**

****