# Mars Rover Kodu Tech Design

This document serves to list the technical work required for adding Mars Rover functionality to Kodu. As the scope is limited and the timeline very short, this document will serve more as a list of features/tasks to be done.

The following sections break down tasks by area:

## Rover Actor

The following code tasks are required to add a functional Rover actor into Kodu:

* Rover FBX added to project
* Rover actor xml added defining rover characteristics
  + Code support for helping art determine surfaces applied
  + Assist design to adjust xml properties to tune desired rover behavior
* Add a special Chassis type to provide characteristics that match the rover’s movement style
* Rover model animation xml defining each animation supported in game
* Code support for rover-specific animations (animations associated with Beam/Inspect/Scan/etc.)
* Code support for rover-specific emitters (i.e. fx associated with the Rover’s beam, etc.)
* Ensure Rover correctly interacts with all pre-existing tiles (actuators, filters, modifiers, etc.)

## Rock Actors

For each rock, we’ll also have the following tasks:

* Rock FBX added to project
* Rock xml added defining each rock type’s characteristics
  + Code support for helping art determine surfaces applied
  + Assist design to adjust xml properties to tune desired rover behavior
* Xml configurations to allow rover actuators to perform on the various rock types
* Functionality to allow switching rock type (i.e. once a scan occurs, “revealing” what type of rock was actually present)
* Support for randomly changing a rock’s score value post reveal
* Code support for rock-specific emitters (i.e. smoke/etc. when hit with a beam)

## New Rover-specific Tiles (Beam/Inspect/Scan)

The following code tasks are required for each rover-specific tile:

* Add actuator’s supporting each new action and configure programming elements to ensure they can only be used with the rover
* Implement each actuator to perform the defined behavior (Beam/Inspect on a single rock, Scan on rocks in the FOV)
* Ensure all combinations of new tile elements are either filtered out or lead to sensible behavior

## New End-of-Path Sensor

We have also been requested to create an End of Path sensor to allow responding to an actor reaching the end of a path it has been following.

* Investigate using the existing WayPoint functionality to determine the desired end of a path an actor is moving along

## Additional Tasks

There are a few tasks in addition to those listed above:

* Make sure LODs properly implemented for all models
* Support for tutorial creation for each of the levels
* Support design with level-building as needed
  + Add new tiles if desired functionality cannot be achieved with the current set
  + Help investigate behavior of existing tiles as needed
* Support art with asset creation as needed
* Check with SCoy on whether the tool for converting from DEM to Kodu is included in the source, and if so, instructions for calling it
* Convert DEMs for Level 4

## Nice-To-Have Tasks

As well, there are a few Nice-To-Have tasks time permitting:

* Improve Rover Chassis to query terrain normal/altitude per wheel and procedurally offset animation bones based on terrain height under each wheel (and reduce clipping with terrain)
  + If procedural update doesn’t work out, another option would be to use additive animations for each wheel