**Underground Game Specification (AVMICO)**

This document specifies the game for the Scirra Underground Jam August/September 2015.

Summary

The purpose of the game is to efficiently mine an area through a serious of semi-autonomous robots. More robots can be purchased using the money raised.

The game is rather like Lemmings in some respects ; objects are sent to a specific position and then they do their thing.

Unlike Lemmings, when they are finished (various definitions !) they return to a queue of objects to be “dispatched”.

Dispatching is done by clicking on the target square then clicking on the object in the queue front window.

There is a smaller button that can be clicked instead of a target square, clicking this pushes the current object at the front to the back of the object queue.

The target square must be (i) visible and (ii) open (e.g. an open area with or without coal, a rail area, or a shaft.

The initial movement direction is given by whether the start position is to the right or left of the shaft centreline. **NOTE:** it can’t go both ways otherwise the search algorithm won’t work.

Moving the window is done via clicking and dragging ; however touching the top line automatically “homes the cursor” so the initial shaft is top and centre.

Game space

The game takes place on a grid of indeterminate size each of which is 48 x 48 pixels (thus fits 13 x 20 squares in a 640 x 960 display). Each square represents an area of the ground. The top line of the grid is grass covered and cannot be mined.

Initially a shaft of depth 10 x 2 wide is predug at the horizontal midpoint of the area (from the top), and the area around this is opened up 3x3 for visibility.

Coal occurs in clumps, Silver and Gold occur in horizontal groups. Diamond occurs singly.

Mapped on top of this is a ‘visibility’ tile map which allows unvisited parts of the mine to be hidden from view, which uses tiles 33 – 38.

The following things can be part of the grid:

|  |  |  |
| --- | --- | --- |
| Code | Description | Description |
| 0 | C:\Construct2\Underground\Graphics\earth.png | Earth, with no coal in it. |
| 1-15 | C:\Users\Paul\AppData\Local\Microsoft\Windows\INetCache\Content.Word\coal7.png | Earth with increasing amounts of coal. |
| 16 | C:\Construct2\Underground\Graphics\empty.png | Empty space (horizontal tunnel) |
| 17-25 | C:\Construct2\Underground\Graphics\empty6.png | Empty space with mined coal, increasing. |
| 26 | C:\Construct2\Underground\Graphics\shaft.png | Vertically dug shaft (Chain) |
| 27 | C:\Construct2\Underground\Graphics\grass.png | Represents the top of the screen |
| 28 | C:\Construct2\Underground\Graphics\rock.png | Rock that cannot be dug through |
| 29 | C:\Construct2\Underground\Graphics\silver.png | Earth containing silver ore |
| 30 | C:\Construct2\Underground\Graphics\gold.png | Earth containing gold ore |
| 31 | C:\Construct2\Underground\Graphics\diamond.png | Earth containing diamonds |
| 32 | C:\Construct2\Underground\Graphics\rail.png | Open space with railway track |
| 33 | C:\Construct2\Underground\Graphics\block0.png | Cover - Open |
| 34 | C:\Construct2\Underground\Graphics\block1.png | Cover - Part open |
| 35 | C:\Construct2\Underground\Graphics\block2.png | Cover - Part open |
| 36 | C:\Construct2\Underground\Graphics\block3.png | Cover - Part open |
| 37 | C:\Construct2\Underground\Graphics\block4.png | Cover - Part open |
| 38 | C:\Construct2\Underground\Graphics\block5.png | Cover - Closed |
| 39 | C:\Construct2\Underground\Graphics\granite.png | Granite. Impenetrable (edge of game) |

Autonomous Vehicles

This is a summary of the AVs in the game

|  |  |  |
| --- | --- | --- |
| Name | Image | Notes |
| Explorer | C:\Users\Paul\AppData\Local\Microsoft\Windows\INetCache\Content.Word\explorer.png | Travels horizontally, reasonably quickly, cannot penetrate anything other than standard earth. Opens 5x5 visual grid rather than 3x3. |
| Miner | C:\Construct2\Underground\Graphics\digger.png | Mines coal, diamonds, gold and silver from adjacent block. When finished moves on to the next block. Cannot penetrate rock. Stores Precious metals and stones but leaves coal on the floor. |
| Driller | C:\Construct2\Underground\Graphics\driller.png | Drills vertically down from the place given. Drills a maximum depth of six squares. Single use only, though can be returned *before* drilling has started. Cannot penetrate rock vertically. Anything penetrated is lost and replaced by the chain. |
| Collector | C:\Construct2\Underground\Graphics\collect0.png | Picks up any coal that is lying about on the ground. Cannot penetrate anything. Automatically returns when the cart is full. The amount of coal carried can be seen visually. |
| Layer | C:\Construct2\Underground\Graphics\layer.png | Lays railway tracks horizontally till it reaches a shaft or wall, cannot penetrate anything. Speed is quadrupled on rails, without rails there is a 75% speed penalty when two carts overlap. |
| Exploder | C:\Construct2\Underground\Graphics\exploder.png | Single use, like driller, can be returned before triggered. Two types of explosion, when in a vertical shaft drills down three squares, replacing whatever is there with chains ; otherwise it extracts coal in a chamber pattern, dropping it on the ground, upwards only, but not removing earth, just the coal…. Okay this doesn’t make logical sense. Rails and Chains are unaffected whatever. |

Note: for the route finder to work, tunnels must always be dug away from the centre, and the shaft must end in a tunnel at the bottom, but not at the top, see the picture. It uses the presence of a shaft to go up, and the absence to go horizontally.

|  |  |  |
| --- | --- | --- |
| C:\Construct2\Underground\Graphics\earth.png | C:\Construct2\Underground\Graphics\earth.png | C:\Construct2\Underground\Graphics\earth.png |
| C:\Construct2\Underground\Graphics\empty.png |  | C:\Construct2\Underground\Graphics\empty.png |
| C:\Construct2\Underground\Graphics\earth.png | C:\Construct2\Underground\Graphics\shaft.png | C:\Construct2\Underground\Graphics\earth.png |
| C:\Construct2\Underground\Graphics\earth.png | C:\Construct2\Underground\Graphics\shaft.png | C:\Construct2\Underground\Graphics\earth.png |
| C:\Construct2\Underground\Graphics\empty.png | C:\Construct2\Underground\Graphics\shaft.png | C:\Construct2\Underground\Graphics\empty.png |
| C:\Construct2\Underground\Graphics\earth.png | C:\Construct2\Underground\Graphics\earth.png | C:\Construct2\Underground\Graphics\earth.png |

TODO List

1. ~~Create levels using a function with global constants for size, create a test page which displays it~~
2. ~~Write a further function which converts that test page into a part-done game.~~
3. ~~Write view opening code.~~
4. ~~Write the ‘search back to the start’ root code and test it.~~
5. ~~Create the initial tile maps and add opening code, and home function and test it.~~
6. ~~Movement via dragging / return to home tapping on top?~~
7. ~~Marker code~~
8. ~~Design the wagon object.~~
9. Write Wagon Object creation code
10. Write code to do OUTBOUND, DONE and RETURN states and test it works.
11. Implement the EXAMINE/WAIT/EXECUTE/MOVE core part.
12. Implement the wagon object functions and test they do what they are supposed to. Each function goes in its own event file.
13. Implement and test the ‘throughput’
14. Implement and test the ‘shop’
15. Title page etc.

Wagon Object

|  |  |
| --- | --- |
| State | Notes |
| WST\_QUEUED | Wagon is in the queue.  Not visible on screen. |
| WST\_OUTBOUND | Triggered by function *WagonStart(UID,x,y)*  Creates a route using the route code, makes visible, and sets init position.  Follows route until route is complete.  Clicking on wagon in this state calls WagonReturn(UID) (e.g. WST\_RETURN)  When route is complete enter WST\_EXAMINE |
| WST\_EXAMINE | Transient state. Wagon dependent  Figures out: what to do, where next move is, how long it takes.  May decide can’t do anything, if so enters WST\_DONE state.  If clicked in this state, goes to WST\_DONE  Otherwise goes to WST\_WAIT |
| WST\_WAIT | Wait state, pauses depending on calculation in WST\_EXAMINE.  If clicked in this state, goes to WST\_DONE.  When complete go to state WST\_EXECUTE |
| WST\_EXECUTE | Transient state. Wagon dependent  Does what was specified in WST\_EXAMINE.  If clicked in this state, goes to WST\_DONE.  If ActionMaxCount > 0, decrement it, and if zero go to WST\_DONE  Otherwise go to WST\_MOVE |
| WST\_MOVE | Move to square occupied.  If clicked in this state, goes to WST\_DONE.  Go to state WST\_EXAMINE |
| WST\_DONE | Dependent on the isOneShot instance variable, the object either self-destructs, or calls WagonReturn() |
| WST\_RETURN | Triggered by calling *WagonReturn(UID)*  Creates a route back to the main shaft  Follows route until route is complete.  When route is complete, pushes UID onto queue and sets state to WST\_QUEUED, becomes invisible again. |

|  |  |  |
| --- | --- | --- |
| Instance | Type | Notes |
| name | String | Name of string (Explorer, Miner etc.) |
| state | Number | Current state |
| isOneShot | Boolean | True if object self-destroys rather than recycles. |
| direction | Number | -1 or 1 depending on the active direction movement of the wagon. |
| viewDistance | Number | Distance view opens up as the object moves. Normally 1, but for Explorer this is 2. |
| actionMaxCount | Number | If this is non-zero, the number of times an action can be done. Exploder is 1, Miner is 6, all others run infinitely. |
| xPos,yPos | Number | Position in tile coordinates. |
| xTarget,yTarget | Number | Where the object is moving towards, in tile coordinates |
| velocity | Number | Speed in tile units / second, normal. |
| routeList | String | List of places to visit as coordinate pairs separated by a vertical bar. |
| coalCount | Number | Units of coal |
| silverCount | Number | Units of silver |
| goldCount | Number | Units of gold |
| diamondCount | Number | Units of diamond |
| waitTime | Number | Time when WST\_WAIT phase elapses, this is when the wagon is ‘working’. |

Additionally there is a global variable “wagonsOperating” which is non-zero when wagons are moving.

The functions Check and Execute further describe the behaviour of the wagon. Note in the examples “XXXX” are used for the wagon name, e.g. this would be Exploder, Miner, Layer and so on and the function would be WagonCheckLayer.

***Function WagonCheckXXXX(UID)***

Check to see if the wagon can operate where it is – can it move, drill, mine, whatever. If it cannot, this function should return -1 which indicates end/go home. If it can, it should return the number of seconds the wagon takes to perform that task – e.g. after that time has elapsed the tunnel will be dug, mine will be drilled etc. Note that some tasks can be done but are effectively “nothing” – e.g. mining a square with nothing mineable, or laying track where it already exists. In this case it should return 1/velocity

***Function WagonExecuteXXXX(UID)***

The wagon should execute the task in question – drill, mine or whatever.

These functions should be in their own event file, under the Wagons subdirectory.