RMORS – RED MOON OVER RISING SUN

A video game design by Ken Poirier

Version 6

Table of Contents

Overview

Summery

In RMORS (pronounced as “remorse”) you are the chief administrator of either China or Japan’s space program in a race to be the first to reach the moon. In this alternative time line story, you’ll be participating in a post-WWII Cold War scenario where the space race was used as a means of avoiding the Korean and Vietnam wars.

Gameplay

RMORS is a resource-based, real-time strategy game in minimalist style. The game will have very few graphics and those it does will have a retro CGA/vector style. RMORS is a fun game for one or more players that consumes little battery power, making it perfect for mobile or online browser play.

Mindset

Set in the Cold War era, the players of RMORS will be intrigued by the story of espionage, propaganda, and nationalism in this retro-scifi.

Inspirational Works

The inspiration for RMORS comes from the following: A Dark Room, Buzz Aldrin’s Race into Space, Lunar Lander, and Space War/Asteroids.

Map/board structure and workflow

LVL 0 = game world

[Game World] LVL 1 = China Complex, Japan Complex, Space, The Moon

[Each Complex] LVL 2 = Admin Office, Garage, Barracks, Research Lab, JPL Lab, Manufactory, Assembly, Launch Pad, Astronaut Training, Radio Array

[Space] LVL 2 = Liftoff, Mission Flight, Re-Entry, Quarantine

[The Moon] LVL 2 = Landing, Surface Exploration, Liftoff, Docking

Players experience the game as well as all interaction takes place at level 2 on the game board.

The Game board can also be represented as:

Game world

\|

China Complex	Japan Complex	Space	Moon
|	|	|	|
Admin Office	Admin Office	Liftoff	Landing
Garage	Garage	Mission Flight	Surface Ex
Barracks	Barracks	Re-Entry	Liftoff
Research Lab	Research Lab	Quarantine	Docking
JPL Lab	JPL Lab		
Manufactory	Manufactory		
Assembly	Assembly		
Launch Pad	Launch Pad		
Astro-Training	Astro-Training		
Radio Array	Radio Array		

Victory Condition

RMORS can be “won” by controlling 4 of 7 landing zones on the moon. Landing zones are accessed through the Mission Flight with the proper PLAN and are controlled by planting flags during the Lunar Exploration gameplay.

Along the way, there are a number of sub-goals and achievements involving exceeding the steps to the moon before your opponent does.

Time

The game progresses through time at an accelerated pace starting January 13th, 1953 and ending at the victory condition by the player or the opponent. There is no “time-out” condition.

The pace of the clock can be controlled by either player. Speed up by paperwork, or slowed down by tea.

The game also has a day/night cycle that can affect features and variables. For example, your Complex building helpers will go home at night and arrive in the morning.

Complexes

Each side of space race, the Player and the Opponent, has their own Space Complex. Japan's Complex is *JapanComplex* and China's is *ChinaComplex*. Each complex is made up of buildings in which the Player or the Opponent can interact to create the resources, equipment, and personal necessary to make the trip to Space [LVL 1] (accessed through the Lanchpad Building), and ultimately, the Moon[LVL 1] (accessed through the Mission Flight).

The player experiences the game as the program administrator in charge of their complex. While at first it may seem as if the Player and the Opponent have no interaction, they will soon unlock the option to engage in propaganda and espionage tactics with one another.

Code

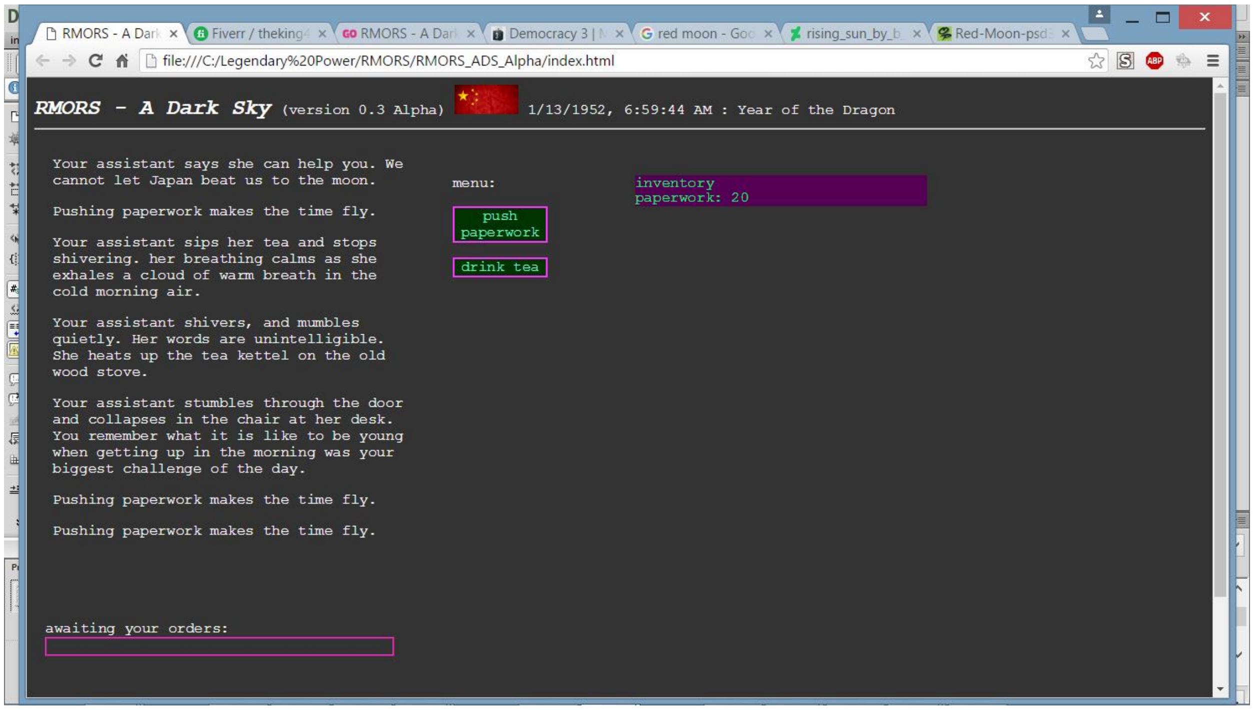
The Complex objects is the main object that the players interacts with. It is essentially their score board and unlocks, hence why each has their own instance.

```
GameWorld{
        //GameWorld object creation
        isRoot = true   //incase anyone wondered
        Parent = null
        // create complexes
        ChinaComplex = create_instance(complex (self, "China", "human", "me")) // parent, country,
        JapanComplex = create_instance(complex (self ,"Japan", "robot", "terminator")) // use ai named terminator
}
        // JavaScript Document

        /*      The Complex Master Object ("class") functions
        *       japanComplex and chinaComplex in the game are derived and prototyped
        *       from this object. The complex instances store the building instances
        *
        *       By Ken Poirier
        */
```

```
const Complex = {
        parent: -1,
        country: "Unknown",
        possesive: "Unknown",
        opponet: "Unknown",
        oPossesive: "Unknown",

        getCountry: function() {
                a[0] = this.country;
                a[1] = this.possesive;
                a[2] = this.opponet;
                a[3] = this.oPossesive;
                return a;
        },
        setCountry: function(key) {
                if (key == 1 || key == "China"){
                        this.country = "China";
                        this.possesive = "Chinese";
                        this.opponet = "Japan";
                        this.oPossesive = "Japanese";
                }
                else if (key == 2 || key == "Japan"){
                        this.country = "Japan";
                        this.possesive = "Japanese";
                        this.opponet = "China";
                        this.oPossesive = "Chinese";
                }
                else {
                        this.country = "Unknown";
                        this.possesive = "Unknown";
                        this.opponet = "Unknown";
                        this.oPossesive = "Unknown";
                        }
                }
        }
```

Buildings

Buildings (LVL 2 on the design schematic) are the areas that the player can explore. Player starts in the Admin office and quickly unlocks the garage which is used to create other buildings. Each building is made up of rooms and stats that allows the player to interact with their or their opponent's complex, and eventually allow them access to Space and The Moon.



RMORS - A Dark Sky (version 0.3 Alpha) 1/13/1952, 6:59:44 AM : Year of the Dragon

```
Your assistant says she can help you. We
cannot let Japan beat us to the moon.

Pushing paperwork makes the time fly.

Your assistant sips her tea and stops
shivering. her breathing calms as she
exhales a cloud of warm breath in the
cold morning air.

Your assistant shivers, and mumbles
quietly. Her words are unintelligible.
She heats up the tea kettel on the old
wood stove.

Your assistant stumbles through the door
and collapses in the chair at her desk.
You remember what it is like to be young
when getting up in the morning was your
biggest challenge of the day.

Pushing paperwork makes the time fly.

Pushing paperwork makes the time fly.

awaiting your orders:
```

Helpers

Each building has an AI character called the helper. The helper provides help/tutorial information, story progression, and automated building management.

Code

```
const Building = {
```

```
parent: -1,

seedRoom: -1,

rooms: [],

aNameRef: 0,

aName: [],


getID: function() {

        return this;

},

getDisplayName: function() {

        return this.aName[this.aNameRef];

},

setDisplayName: function(s, x) {

        this.aName[x] = s;

},

setDefaultName: function(s) {

        this.defaultName = s;

},

getPop: function()  {

        return this.population;

},

getMaxPop: function() {

        return this.maxPop;

},

getPopSpace: function() {

        return (this.maxPop - this.population);

},

peeper: function(i){

        switch (i) {


                case 0:
```

```
                return this.look;
                break;
                case 1:
                return this.sLook;
                break;
                case 2:
                return this.vbLook;
                break;
                case 3:
                return this.altLook;
                break;
                default:
                return false;
        }
},
setLook: function(s, i) {

        //if (i == 0) this.look = s;
        switch (i) {

                case 0:
                this.look = s;
                break;
                case 1:
                this.sLook = s;
                break;
                case 2:
                this.vbLook = s;
                break;
                case 3:
                this.altLook = s;
                break;
                default:
                return false;

            }

            return true;

        }
}
```

Rooms

As of version 0.09 Alpha, buildings are made up of rooms (previously called upgrades), which is the point of interaction for the player. These rooms can be traversed in the normal cardinal directions of traditional Text Adventures. As players upgrade their building, it creates a new room by attaching it to an existing room in the building. The first room of the building is referred to as the “Seed Room”.

```
    // JavaScript Document

    /*      "Room" Master Object ("class") functions
    *       All rooms in the game are derived and prototyped
    *       from this object.
    *       Rooms are a child class of buildings and the player
    *       can move from room to room in the building.
    *
    *       room2.js is a json alternative Room object to room.js (don't use both)
    *
    *       By Ken Poirier
    */

    const Room = {
            name: "unamedRoom",
disName: "Unnamed Room",
storyName: "an unammed room",
curPop: 0,
maxPop: 0,
//parent: false, //use buildID instead
buildID: -1,
roomItems: [],
visits: 0,

//conectors and movement
isSeedRoom: false,
Nroom: false,
NEroom: false,
Eroom: false,
SEroom: false,
Sroom: false,
SWroom: false,
Wroom: false,
NWroom: false,
Up: false,
Down: false,
secretPassage: false,
buildingExit: false,
buildingLevel: 1, //what floor are we on

//ways to get up down
ladder: false,
stairs: false,
rope: false,
lift: false,
pole: false,
shaft: false,
//Qualities

lighting: "dark",

smell: "nothing",

abientSound: "quite",

inside: true,

underground: false,


//Look text

fLook: "You look at the empty room for the FIRST time.",

sLook: "It is still just an empty room.",

vLook: "It is a very empty room. There is nothing here. Nothing at all.",

altLook: "You look at the empty room and notice something you did not see before.",


//functions


getID: function() {

        return this;

},


travel: function(compass) {

        if (canAddRoomLink(compass)){

                errorPrint("You can't go that way.");

                if (info.debug){

                        console.log("Player tried to go " + compass + " but nothing is there");

                }

        }

},


teleport: function(ID) {


        if (ID  != null){
                errorPrint("You attempt to fast travel to a new location but it doesn't exist.");
                if (info.debug){
                        console.log("teleport failure");
                }
        }
        else {
                info.lRoomID = this.getID;
                info.cRoomID = ID;
                info.hasTraveled = true;
        }
},

getDisplayName: function() {
        return this.disName;
},

getResponse: function(input) {
        return "responses not setup for this room.";
},



getVisits: function() {
        return this.visits;
},

addVisit: function() {
        this.visits = this.visits +1;
        return "visit number " + this.getVisits() + " added";
},

firstLook: function() {
        return this.fLook;
```

```
},

lookRoom: function() {
        if (info.verboseOn) {
                return this.vLook;
        }
        else { return this.sLook; }
},

vLookRoom: function() {
        return this.vLook;
},

getID: function() {
        return this;
},

canAddPop: function() {
        return(this.curPop < this.maxPop);
},

canAddRoomLink: function(s) {
        sc = s.toUpperCase();
        if (sc = "N"){
                if ( !this.Nroom){
                        return true;
                }
        }
        if (sc = "NE"){
                if ( !this.NEroom){
                        return true;
                }
}
if (sc = "E"){
        if ( !this.Eroom){
                return true;
        }
}
if (sc = "SE"){
        if ( !this.SEroom){
                return true;
        }
}
if (sc = "S"){
        if ( !this.Sroom){
                return true;
        }
}
if (sc = "SW"){
        if ( !this.SWroom){
                return true;
        }
}
if (sc = "W"){
        if ( !this.Wroom){
                return true;
        }
}
if (sc = "NW"){
        if ( !this.NWroom){
                return true;
        }
}
return false;
```

```
        },


        getNextFreeRoom: function() {
                if (canAddRoomLink(N)) return "N";
                if (canAddRoomLink(E)) return "E";
                if (canAddRoomLink(W)) return "W";
                if (canAddRoomLink(S)) return "S";
                if (canAddRoomLink(NE)) return "NE";
                if (canAddRoomLink(SE)) return "SE";
                if (canAddRoomLink(SW)) return "SW";
                if (canAddRoomLink(NW)) return "NW";
                return "No room connections available";
        }



}
```

Admin Office

This is the first building you start off in. The Admin Office is the player's source of information.

The Assistant

The player is provided with an assistant. The assistant arrives in the morning and leaves at night.

The assistant can provide hints, or be assigned tasks.

Possible future AI social interactions.

PPb - Push Paperwork Button

This is the first building you start off in. It contains the "push paperwork button" (PPb). The PPb raises a constantly lowering meter called the PPm, which has a range of 0.1f to 1.0f. The lower the PPm the slower time advances, the higher the PPm the quicker time advances. The feature could be useful in multiplayer mode too.

Drink Tea

The player can drink tea made by their assistant. Counter to the PPb, drinking tea slows the game down.

Building Population

The Admin office has a base population of ¼. 1 is the minimum population of the admin office and represents the player. The population is expandable by upgrading the Administration office. New administrators appear randomly when the population is below the maximum.

Buttons

- Push Paper Work
- Get Mail
- Check the Budget (appears by Event)
- Audit Space Program (requires 2 administrators)
- Plan a Mission (requires Launch Pad + Assembly + Manufactory)
- Upgrade Administrative Office (requires Garage + barracks + resources)
- Change Building/Travel (requires Garage)

Events

The PPb also creates an increased chance of a random event happening. Events are different depending on which complex or building you are in. Pushing the PPb gives a textout that is a witty glib about office work. A low PPm could return a comment about how much work needs to be done, or a high PPm could likewise be reflected in a text out.

Example Events-

Name: Budget Approval

Condition: 4+ PPb Clicks and Check Budget locked

Effect: Unlocks Check Budget

Textout: The Budget for the space program has been approved by the parliament. You can now track your spending at any time.

Name: Garage Open

Condition: PPm > 0.8f and Garage is locked

Effect: Garage unlocked

Textout: The garage is now open. The men are ready to get to work.

Resources

- Paperwork – Paper work is a useless resource. It is generated by the depletion of the PPm. The Admin office starts off with 3,000 Paperwork and pushing the PPb subtracts 10. It is there to give the player a sense of satisfaction, and to remind them to push the PPb.
- Currency – is acquired after each budget cycle from the budget committee or by completing objectives in the Budget Screen.

Get Mail

Soon after pushing the paperwork you have the ability to get the mail once a day, which gives random amounts of paperwork, currency, and other goodies. Building the mailroom upgrade to the Admin Office will bring in more items.

Budget Screen

The Budget Screen displays the current funds, project future funds, spending, and bonus objectives. Players receive a monthly stipend of currency from their government based on their reputation. Reputation can be increased or decreased based on mission success and failure as well as by their interactions with the press.

Having excessive amounts of paperwork (greater than 20,000), results in a downgrade in reputation, and thus a smaller budget stipend.

Audit Screen

Displays Score information, statistics, and info on who is winning the space race.

Planning a Mission

Set up a mission of preset parameters with currently researched equipment (see Research and JPL labs), called the PLAN.

The PLAN is sent to the manufactory building and non-existing PARTS are built and sent to the assembly along with the PLAN.

The Assembly assembles the PARTS in to a ROCKET that is sent to the launch pad along with the PLAN.

From the Launch Pad, the mission can be launched.

The PLAN can be scrubbed at any time from any above facility.

```
PLAN{
        //base parameters//
        Mission=”suborbital unmanned gemini”
        planNum = 2    // a quick id of mission objectives
        MissionNum = 12  //
        isSrcubbed = false
        awaitsOrders= true
        isRocket = false

        //PARTS//
        capsule=0
        fuel=0
        engine=0
        bosters=0
eva=0


//Needed PARTS

//each INT represents a type of part, not a quantity

capsuleNeeded=2

stage[1] = 1

stage[2] = 0

stage[3] = 0

fuel=1

engine=1

bosters=0

eva=0


//Crew

crewNeeded = 0

Astronaut[0] = null

…

}
```

Upgrades

The Admin office can be upgraded to increase population and assign focused duties such as paperwork mill, mail room, budget office, and auditing office, thus providing more information in the appropriate reports.

Garage

The garage is the second building the player receives in the game. It is a free building. This building receives supplies and allows the construction of other buildings.

Population

The base population of the garage is 4/6, 3 workers and 1 union boss (see Types of Workers). Population can increase by upgrading, same as admin building. 1 worker performs 1 work per tick in game time. If too much work is backed up, workers will go on strike, in which they will only perform work on the garage.

The garage comes two base upgrades: 1 Cargo Bay and 1 Storeroom

Buttons

- Talk to Union Boss
- Build a New Building
- Check Receivables
- Review Work Orders
- Upgrade Garage (available after barracks built)
- Change Building

Talk to Union Foreman

Talking to the foreman give a status report on the happiness of the workers and gives tutorial like advice as a textout.

Build New Buildings

Creating a new building requires resources. With the right resources the player can build other buildings. Once the player chooses to construct a building it is added to the WORK ORDER queue. Each building has a designated number of build points needed to make the building active and usable. Workers will only work on the project on the top of that queue. New buildings selected to be built go to the bottom of the WORK ORDER queue.

Choosing to add a new building to the WORK ORDER queue consumes the needed resources.

Resources

- Steel
- Concrete
- Glass
- Conduit
- Wires
- circuits
- Microchips
- Plastic
- Space Age Material

Check Receivables

This button will bring in a number of resources from the above list. There is a refectory period in between button presses. The more cargo bay upgrades, the more resources acquired and better chances of a high drop, the more storage room upgrades, the shorter the refectory period between presses.

Review Work Order

The WORK ORDER queue is a queue of new buildings and upgrades that need building. Here that list can be rearranged as need. Rearranging the order is not resource penalized, but instead too much rearranging can push the workers towards a strike. Workers will only work on the project on the top of the queue, but moving a build order down the list does not negate progress. Building projects cannot be canceled.

Upgrades

Name: Cargo bay

Cost: 500 concrete, 50*x currency (x=# of cargobays)

Function: Increases Resource drop quality

Name: Supply Room

Cost: 50 steel, 75*x currency (x=# of Supply rooms)

Function: reduces refraction time between resource drops

Name: Breakroom

Cost: 50 concrete, 25 steel, 100*x currency (…)

Function: increase worker population cap by 6

Types of workers

Of the total population of workers, they may mutate into other types of workers

- Worker – default does 1 build point per game tick
- Striker – does no work
- Union boss – six strikers can create a union boss
- Expert Worker – does 3 build points per game tick (created by intern)
- Master worker – does 5 build points per GT (created by expert workers)
- Saboteur – can sabotage opponent's complex (created by radio array)

Barracks

The barracks house scientists, astronauts, interns, and press agents.

Astronauts and scientists live in the barracks but will only work if the respective facilities are available.

Interns increase the chance of a worker or scientist becoming and expert.

Press agents and their favorable opinion help increase the complex's currency acquisition.

Research Lab and JPL

The labs where scientists work to unlock technologies on a civ style technology tree. Scientists working in the research Lab produce Science Points (SP) which unlock technology tree and Scientists in the JPL produce Jet Science Points (JSP) which are used to develop prototype items.

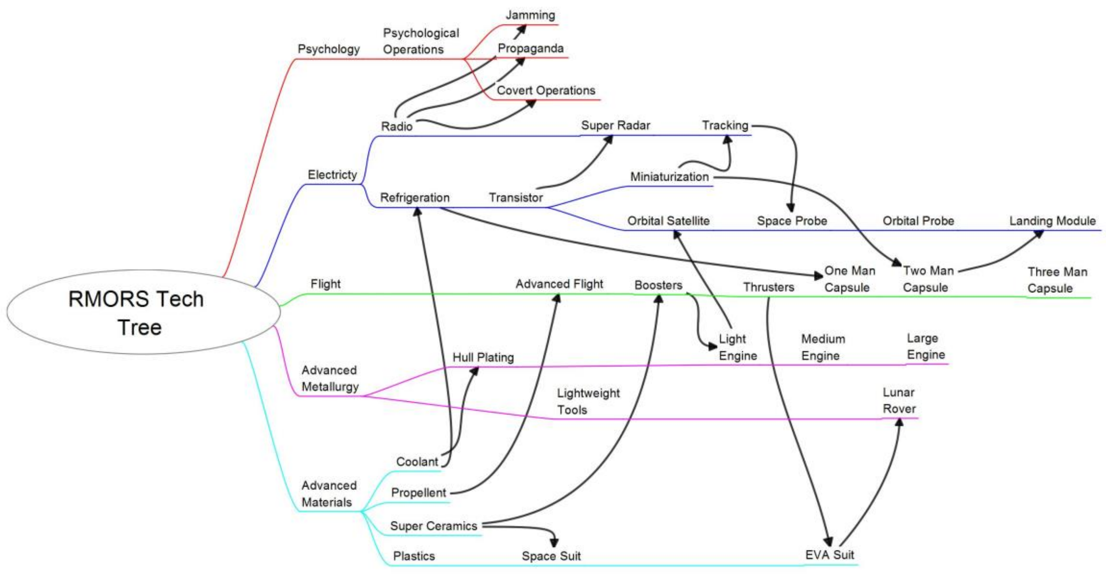
The JPL require **Advanced Flight** technology to be fully researched.

Scientists can work at either facility. But they are a bit lazy. Unlike workers, scientists won't go on strike, but they will have a work slowdown if there are more science jobs than there are scientists, so you'll want to have more in the barracks then you need.

Scientist mutations

- Scientist = default, can produce 1 SP or 1 JSP when working
- Mad Scientist = if there are 5 scientists not working and no mad scientist (so 10 not working to produce 2), one will mutate into a mad scientist, mad scientists produce SP or JSP to random projects constantly, when they are in the lab however, they produce no additional points towards the currently selected research project.
- Dedicated Scientist = Mutation caused by interns, dedicated scientists produce 2 SP or 2 JSP when working. Cannot become a mad scientist.
- Reverse Engineer = Created by Radio Array, Reverse engineers produce 3 SP or 3 JSP on any technology the opponent has master and the player has not.
- Expert Scientist = if there are five dedicated scientists and no Expert, one will mutate into an expert. Any scientist working in the same lab room upgrade as an expert scientist get +1 SP or +1 JP while working. Cannot become a mad scientist.
- Astronaut scientists = A former astronaut that washed out of the program and is converted to a scientist that does 2 SP or 2 JSP. Cannot become an expert or mad scientist.

The technology tree



Starting Tech

Players start the game with Electricity, Radio, and Flight. The rest must be unlocked using SP created by the Laboratory.

Manufactory

Once scientists have developed the technology, the manufactory can start producing spaceship parts.

Astronaut complex

Unlike other personalities in the game, Astronauts (NAUTs) have their own RPG style stats as they are the ones who go on missions.

Code

```
NAUT{
        Name = "Key Hu"
        Rank = 0   // 0 = private, 1 = lt, ect
        Exp = 0
        Status = 1 // 0= dead 1 = active 2 = injured 3 = sick
        isOnMission = false //currently in SPACE or MOON
        isSetForLaunch = false  // part of a future mission
        //Stats range from 1 to 5
        Strength = 1
        Agility = 2
        Endurance = 2
        HP = Endurance * 10
        Intelligence = 1
        Pyche = 5   //mental health
        //skills range from 1 to 5
         Pilot = 3
        Docking = 0
        LunarLander = 0
        EVA = 0
}
```

Upgrades

Upgrades to the astronaut Training center allow the training of stats and skills, as well as let players practice mini-games and scenarios they will encounter in space.

NAUT health

Injuries, sickness, and bad missions can subtract endurance. If endurance reaches zero, the NAUT will washout and, if high enough in rank can become an astronaut scientist, else the NAUT is dismissed.

Assembly Building

Here spaceship parts are made into rockets. Once a rocket is assembled it can be sent to storage or sent to the launch pad.

Launch Pad

Here the player can see all the information on their mission, weather conditions, their rocket, and the NAUTS selected for the mission.

When ready, the player can launch the mission starting the SPACE (lvl 1 on board workflow).

Upgrades

- Pad Expansion – the launch pad can be expanded to launch up to three ROCKETs simultaneously.
- Fire Proofing – improves launch safety 15%

Radio Array

The radio array is required to communicate with NAUTS in space and track objects in the sky.

It is also used to JAM the opponent’s communications and to spy on the opponent. Here you can also create spies, such as the Saboteur and the Reverse Engineer, using the secret room upgrade.

Upgrades

- Secret Room – Train spies
- Radio Dish – Up to 20 dishes can be built. Can be set to spy, jam, or transmit.

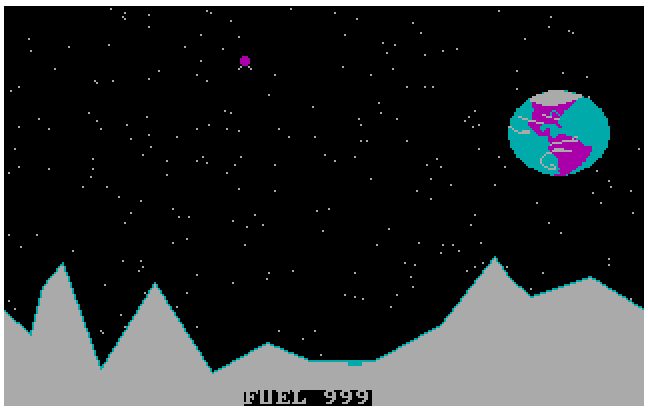
The Moon

The Moon, like space, is affected by both the player and their opponent. The moon's surface is represented as a 2D top-down style map and players move in a turn based style (think Ultima I-III). The map is revealed through the LUNAR FLYBY and Lunar Satiate.

Once the player has adequately assessed a landing site, and have the means, they can land on the moon and explore. There are seven landing sites and the first player to explore four of them wins the game.

Lunar Landing

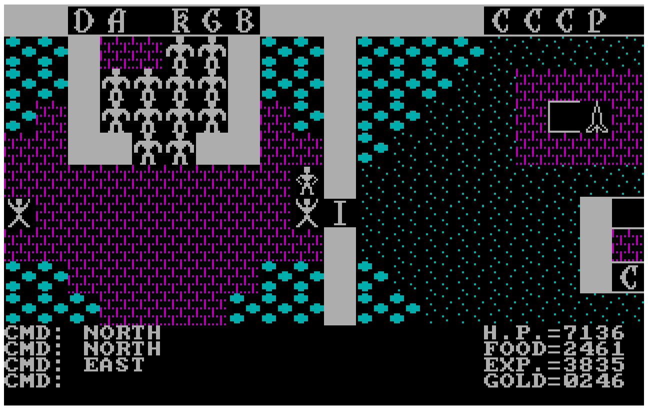
This board is essentially the classic arcade game of Lunar Lander with extra saving throws depending on the quality of the player's Landing Module and NAUTS.



Moon Lander (1986)

Lunar Exploration

The moon's surface is represented as a 2D top-down style map and players move in a turn based style (think Ultima I-III).



```
CMD: NORTH          H.P.=7136
CMD: NORTH          FOOD=2461
CMD: EAST           EXP.=3835
CMD:                GOLD=0246
```

Ultima 2 CGA

It is in this portion of the game, players can achieve the final victory condition. The first player to capture 4 of 7 landing zones is the winner. Landing zones are captured by exploring the landing zone and placing 3 of 5 flags and returning their NAUTs back to earth successfully.

Both players can occupy the same landing zone at the same time and compete for flags, but once a flag is captured, it cannot be changed.

Lunar Liftoff

Like Lunar landing but in reverse, players must lift off and orbit with the capsule.

Docking

After lunar lift-off, the player must dock the LM with the capsule to return to space. In this mini-game, the player must move the dot to the center of circle, or make the X Y coordinates equal to zero (for visually impaired mode).

TEXTOUT:

Left more!

Down more!

Left more!

Appendix A – Resources

Admin Building

- Currency – is acquired after each budget cycle from the budget committee or by completing objectives in the Budget Screen.
- Paperwork – Paper work is a useless resource. It is generated by the depletion of the PPm. The Admin office starts off with 3,000 Paperwork and pushing the PPb subtracts 10. It is there to give the player a sense of satisfaction, and to remind them to push the PPb.

Building Materials

- Steel
- Concrete
- Glass
- Conduit
- Wires
- circuits
- Microchips
- Plastic
- Space Age Material

Appendix B - Glossary of terms

ADR – A Dark Room

ADS – A Dark Sky

AI – Artificial Intelligence. Characters in RMORS use machine-state style AI.

Browser Based – A Browser based game is a game played in a web browser such as chrome, fire fox, or IE.

Capsule – A special payload that hold NUATs.

Console – Where users can type textual input

Currency – The Yuan in china. The Yen in Japan.

Engine (game) – the sum of all code that runs the game.

Engine (rocket) – a part of a rocket that expends fuel for movement.

Event – a part of the game that is non-standard play for the current game board, like an aside in Shakespearean terms. Events have their own game board, often involving answering a series of questions, solving a puzzle, or engaging in combat.

Flag – a marker on the moon to signify capturing an area for the victory condition.

Fuel Tank – Holds x amount of fuel for space missions, used by engine (rocket).

Lunar Module – a special vehicle for landing and taking off from the moon.

NAUT – a space traveler, an astronaut

Parser – A program that allows the computer to read and understand text, particularly input from the console.

PLAN – a mission plan: from assembly, to launch, to space, to moon, back to space, and landing.

Payload – Either a satellite, NAUT capsule, lunar module, or dummy payload.

PPb – Push Paperwork Button.

PPm – Push Paperwork meter controls the flow of time in the game. Pushing the PPb speeds the clock up and drinking Tea slows it down.

RMORS – acronym for “Red Moon Over Rising Sun” (pronounced as remorse)

Rocket – a space ship consisting of (at minimum) an engine, fuel tank, and a payload

Victory Condition – the set of circumstances for which one can win the game.

Appendix C – A Dark Sky

Overview

A Dark Sky (ADS) is an online-browser-based version of RMORS that is heavily influenced by A Dark Room by Doublespeak. In ADS the players will run the Chinese campaign against a Japanese opponent simulated by clock events.

This prototype version of RMORS may have redacted features for the sake of rapid development. The features and mechanics of the complete RMORS may change upon discoveries during this prototyping process.

In order to help engage the community the source code for ADS is openly available to the public to compare with the source code to A Dark Room. Also a command-line text parser is intergraded into the game to make it more accessible to visually impaired/disabled players and to add another dimension of world exploration.

Code

Index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>RMORS - A Dark Sky</title>

<link href="css/ADAA.css" rel="stylesheet" type="text/css" />


	<script>

	  (function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){

	  (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),

	  m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)

	  })(window,document,'script','https://www.google-analytics.com/analytics.js','ga');
```

```
            ga('create', 'UA-55642323-4', 'auto');

            ga('send', 'pageview');


          </script>


  <!-- Standard Libraries -->


  <script type="text/javascript" src="lib\jquery2.js"></script>


  <!-- Strings -->

  <script type="text/javascript" src="strings/adminStrings.js"></script>


  <!-- Game Source -->

  <script type="text/javascript" src="scripts/getpet.js"></script>

  <script type="text/javascript" src="scripts/clock.js"></script>

  <script type="text/javascript" src="scripts/resources.js"></script>

  <script type="text/javascript" src="scripts/storyReader.js"></script>

  <script type="text/javascript" src="scripts/buttons/button.js"></script>


  <!-- Base Objects -->

  <script type="text/javascript" src="scripts/buildings/rooms/room2.js"></script>

  <script type="text/javascript" src="scripts/buildings/building2.js"></script>

  <script type="text/javascript" src="scripts/roomItems/roomItem2.js"></script>


   <!-- Items -->


  <script type="text/javascript" src="scripts/roomItems/adminItems.js"></script>




  <!--Characters-->
```

```
    <script type="text/javascript" src="scripts/characters/helper.js"></script>


    <!-- Rooms -->
    <script type="text/javascript" src="scripts/buildings/rooms/admin/darkRoom.js"></script>
    <script type="text/javascript" src="scripts/buildings/rooms/admin/mainOffice.js"></script>
    <!-- Buildings -->


    <script type="text/javascript" src="scripts/buildings/admin2.js"></script>
    <!--Complexs-->
    <script type="text/javascript" src="scripts/buildings/complex2.js"></script>





    <!-- Buttons -->




    <!-- game world variables -->
    <script type="text/javascript" src="scripts/info.js"></script>








</head>


<body>


<!--Audio Ids-->


<audio id="snd_light" src="sounds/lights.wav"></audio>
<audio id="snd_ppb" src="sounds/ppb.wav"></audio>
<audio id="snd_ready" src="sounds/ready.wav"></audio>
```

```
<audio id="snd_tea" src="sounds/tea.wav"></audio>

<audio id="snd_build" src="sounds/Building.wav"></audio>

<audio id="snd_wait" src="sounds/wait.wav"></audio>


<header>

<table width="100%" border="0" cellspacing="0" cellpadding="0" id="topmenu">

  <tr>

    <td width="28%"><a id="top"><a href="http://www.kenpoirier.com" title="Home" target="_self">KenPoirier.com</a></a></td>

    <td width="18%"><a href="http://www.kenpoirier.com/science" title="coming soon" target="_self">Science</a></td>

    <td width="15%"><a href="http://www.kenpoirier.com/art" title="my art" target="_self">Art</a></td>

    <td width="22%"><a href="http://kenpoirier.com/gaming" title="I make video games" target="_self">Gaming</a></td>

    <td width="17%">&nbsp;</td>

  </tr>

</table>

</header>

<br />


    <div id="vTitle"><span class="gameName">RMORS - A Dark Sky</span> <span id="versionInfo"> (version x.xx Alpha)</span> <img src="images/cFlagLong.jpg" width="70" /> <span class="clock" id="time"> </span> ( score: <span id="score">00000</span> )</div>

    <hr />


    <div id="console" class="wrapper">

                <div class="navigation"></div>

        <div class="room">

                <div id="storyLog" class="story"></div>

            <div id="menuTop" class="menu">actions<br /><hr /></div>

            <div id="lookMenu">look at<br /><hr /></div>

            <!--<div class="clock" id="time">tick</div>-->

            <div id="location"><span id="menuName">Location Name</span><hr />

                <div id="upgrades"></div>

            </div>
```

```
      <div id="invTop" class="inventory">inventory <br /><hr /></div>


    </div>
  </div>


<div class="coverbox"></div>


<form onsubmit="return false;" id="consoleBox">
          awaiting your orders:<br />
          <input type="text" size="50" autofocus="autofocus" id="cInput" />


</form>


<p id="message_help" class="story" style="display: none;">No one can help you now...</p>


<div id="faux-terminal">
  <div class="layer"></div>
  <div class="overlay"></div>
</div>



<!--start the game. This code last -->


<script type="text/javascript" src="scripts/start.js"></script>
<style></style>


<br />


</body>
</html>
```

StoryReader.JS

```
// JavaScript Document


/* RMORS - A Dark Sky
*               Story Reader Script
*               Global Functions for text parsing
*               and reporting
*                                       by Ken Poirier
*/


//response writer
//Outputs text to the storybox
function writeReponse (input, response){
                $(document).ready(function() {

                                if ((input != "help") && (input != "info") && (input != "")) {

                                                response = info.cRoomID.getResponse(input, info.cBuildingID);
                                                if (response != "invalid input") {
                                                                $(storyPrint(response)).hide().insertAfter("#storyLog").fadeIn(2000);
                                                }
                                                else if (response == "invalid input") {
                                                $("<p class=\"error\">I don\'t understand \"" + input
+"\".</p>").hide().insertAfter("#storyLog").fadeIn(2000);
                                                }

                                }

                                else if (input == "help") {
                                                $("#message_help").clone().hide().insertAfter("#storyLog").fadeIn(2000);
```

```
				}

				else if (input == "info" || input == "") {

						$(string2story("Game Info:<br />version# " + info.version + "<br />by " + info.author
)).clone().hide().insertAfter("#storyLog").fadeIn(2000);

				}


				/*else if (response == "invalid input") {

						$("<p class=\"story\">I don\'t understand \"" + input
+"\".</p>").hide().insertAfter("#storyLog").fadeIn(2000);

				}*/

				else {

						$("<p class=\"error\">Error: unreachable condition</p>").hide().insertAfter("#storyLog").fadeIn(2000);


				}

		});

}


//response writer

//Outputs text to the storybox old version

function writeReponse2 (input, response){

		$(document).ready(function() {

				if ((input != "help") && (input != "info")) {

						if (response != "invalid input") {

								//$("<p class=\"story\">" + response + "</p>").hide().insertAfter("#storyLog").fadeIn(2000);

								$(storyPrint(response)).hide().insertAfter("#storyLog").fadeIn(2000);

						}

						else if (response == "invalid input") {

						$("<p class=\"error\">I don\'t understand \"" + input
+"\".</p>").hide().insertAfter("#storyLog").fadeIn(2000);

						}

				}

				else if (input == "help") {
```

```
                    $("#message_help").clone().hide().insertAfter("#storyLog").fadeIn(2000);

            }

            else if (input == "info" || input == "") {

                    $(string2story("Game Info:<br />version# " + info.version + "<br />by " + info.author
)).clone().hide().insertAfter("#storyLog").fadeIn(2000);

            }

    });

}


//response writer

//Outputs text to the storybox old version

function writeReponse3 (input, response){

    $(document).ready(function() {

            if ((input != "help") && (input != "info") && (input != "")) {


                    response = info.cBuildingID.getResponse(input, info.cBuildingID);

                    if (response != "invalid input") {

                            $("<p class=\"story\">" + response + "</p>").hide().insertAfter("#storyLog").fadeIn(2000);

                    }

                    else if (response == "invalid input") {

                    $("<p class=\"story\">I don\'t understand \"" + input
+"\".</p>").hide().insertAfter("#storyLog").fadeIn(2000);

                    }



            }



            else if (input == "help") {

                    $("#message_help").clone().hide().insertAfter("#storyLog").fadeIn(2000);

            }

            else if (input == "info" || input == "") {

                    $(string2story("Game Info:<br />version# " + info.version + "<br />by " + info.author
)).clone().hide().insertAfter("#storyLog").fadeIn(2000);
```

```
                }

                /*else if (response == "invalid input") {
                        $("<p class=\"story\">I don\'t understand \"" + input
+"\".</p>").hide().insertAfter("#storyLog").fadeIn(2000);
                }*/
                else {
                        $("<p class=\"story\">Error: unreachable condition</p>").hide().insertAfter("#storyLog").fadeIn(2000);

                }
        });
}

//story output
function storyPrint(s){
        //Console layout
$(document).ready(function() {

        $(string2story(s)).hide().insertAfter("#storyLog").fadeIn(2000);

        });
}

function errorPrint(s){
        //Console layout
$(document).ready(function() {

        $(string2error(s)).hide().insertAfter("#storyLog").fadeIn(2000);

        });
}
```

```
function infoPrint(s){
		//Console layout
$(document).ready(function() {

		$(string2GI(s)).hide().insertAfter("#storyLog").fadeIn(2000);

		});
}

//turns string into storyLog output
function string2story(s){

		return "<p class=\"story\">" + s + "</p>";

}

//turns string into error output
function string2error(s){

		return "<p class=\"error\">" + s + "</p>";

}

//turns string into Game Info output
function string2GI(s){

		return "<p class=\"gameInfo\">" + s + "</p>";

}
```

```
//Parse input and return object manimpulation info

//unlike lookInfo(), ObjMan() first looks for the object and then the verb to go

//with it. returns [ isManipulation?bool, "Object", "verb"]

function objMan(ds){

        //init

        s = cleanString(ds);

        var oma = [ false, "nothing", "none"]; //default


        //check for object

        if (s.search(/(tea).*/) >= 0) oma[1] = "tea";

        if (s.search(/(kettle).*/) >= 0) oma[1] = "kettle";

        if (s.search(/(window).*/) >= 0) oma[1] = "window";

        if (s.search(/(paper).*/) >= 0) oma[1] = "paper";

        if (s.search(/(paperwork).*/) >= 0) oma[1] = "paperwork";

        if (s.search(/(assistant).*/) >= 0) oma[1] = "assistant";

        if (s.search(/(flag).*/) >= 0) oma[1] = "flag";

        if (s.search(/(mail).*/) >= 0) oma[1] = "mail";

        if (s.search(/(radio).*/) >= 0) oma[1] = "radio";

        if (s.search(/(light).*/) >= 0 ||

                s.search(/(lamp).*/) >= 0) oma[1] = "lamp";



        //check for verb

        if (s.search(/(get ).*/) >= 0) oma[2] = "get";

        if (s.search(/(check ).*/) >= 0) oma[2] = "check";

        if (s.search(/(make ).*/) >= 0) oma[2] = "make";

        if (s.search(/(open ).*/) >= 0) oma[2] = "open";

        if (s.search(/(shut ).*/) >= 0) oma[2] = "shut";

        if (s.search(/(push ).*/) >= 0) oma[2] = "push";

        if (s.search(/(pull ).*/) >= 0) oma[2] = "pull";

        if (s.search(/(drink ).*/) >= 0) oma[2] = "drink";
```

```
    if (s.search(/(eat ).*/) >= 0) oma[2] = "eat";

    if (s.search(/(use ).*/) >= 0) oma[2] = "use";

    if (s.search(/(do ).*/) >= 0) oma[2] = "do";

    if (s.search(/(train ).*/) >= 0) oma[2] = "train";

    if (s.search(/(pour).*/) >= 0) oma[2] = "pour";

    if (s.search(/(turn ).*(on).*/) >= 0) oma[2] = "turn on";

    if (s.search(/(turn ).*(off).*/) >= 0) oma[2] = "turn off";

    if ((s.search(/(i ).*(light ).*/) >= 0) || (s.search(/(to light ).*/) >= 0)

        || (s.search(/(burn ).*/) >= 0) || (s.search(/(set ).*( on fire).*/) >= 0)

        || (s.search(/(light ).*/) >= 0 && s.indexOf("light") == 0))

            oma[2] = "burn";


    oma[0] = (oma[2]  != "none");


    if (info.debug) console.log(ds + " -> oma: " + oma);


    return oma;

}


//Parse input and return look at information [,,,Look type]

function lookInfo(ds){

    //init

    var debug = false;

    s = cleanString(ds);

    var lai = [];

    //var lookType = 0; [redacted] no lai[3]


    //what kind of look

    lai[2] = 0; //by default not looking

    if (s.search(/(look)/) >= 0){

        lai[2] = 1;  //is a look
```

```
}

if (s.search(/(look ).*(under )/) >= 0){

        lai[2] = 3; //is a "look under"

}

if ((s.search(/(look ).*(through )/) >= 0) ||

                (s.search(/(look ).*(thru )/) >= 0)

                ){

        lai[2] = 4; //is a "look through"

}

if ((s.search(/(look ).*(out )/) >= 0)){

        lai[2] = 5; //is a "look out"

}

if ((s.search(/(look ).*(in )/) >= 0)){

        lai[2] = 6; //is a "look out"

}

if ((s.search(/(look ).*(over )/) >= 0)){

        lai[2] = 7; //is a "look out"

}

if (s.search(/(look ).*(at )/) >= 0){

        lai[2] = 2; //is a "look at"

}




//lai[0] = (s.search(/(look ).*(at )/) >= 0); //does it exist

lai[0] = (lai[2] > 0); // are we even looking at something?

if (lai[0]){

        lai[1] = "unknown look type";

        if (lai[2] == 1){ lai[1] = "room";} //regular look

        if (lai[2] == 2){

                laIndex = s.search(/(look ).*(at )/); //get start index
                laPhrase = (s.match(/(look ).*(at )/))[0].length; //get size

                /*if (debug){

                        console.log(laPhrase = (s.match(/(look ).*(at )/))[0].length());

                }*/ //commented out for execution speed

                targetIndex = laIndex + laPhrase;

                lai[1] = s.substring(targetIndex); //the target of look at.

        }

        if (lai[2] == 3){

                laIndex = s.search(/(look ).*(under )/); //get start index

                laPhrase = (s.match(/(look ).*(under )/))[0].length; //get size

                /*if (debug){

                        console.log(laPhrase = (s.match(/(look ).*(at )/))[0].length());

                }*/ //commented out for execution speed

                targetIndex = laIndex + laPhrase;

                lai[1] = s.substring(targetIndex); //the target of look at.

        }

        if (lai[2] == 5){

                laIndex = s.search(/(look ).*(out )/); //get start index

                laPhrase = (s.match(/(look ).*(out )/))[0].length; //get size

                /*if (debug){

                        console.log(laPhrase = (s.match(/(look ).*(at )/))[0].length());

                }*/ //commented out for execution speed

                targetIndex = laIndex + laPhrase;

                lai[1] = s.substring(targetIndex); //the target of look at.

        }




}

else{

        lai[1] = "none"
        }

        return lai;


}




function cleanString(s){

        var s1 = s.replace(/\./g, '');

        var s2 = s1.replace(/,/g, '');

        var s3 = s1.replace(/\!/g, '');

        return s3.replace(/\?/g, '');

}
```

Button.JS

```
        // JavaScript Document

        //button template/object class


        var Button = function (lable, refract, locationID, command, htmlID) {

                this.label = lable;        //what does it say on the button
                this.refract = refract;  //if you press it, how long until it can be pressed again?
                this.rTimer = 0; //time left in refraction period
                this.locationID = locationID; //what building/room is this button in?
                this.command = command;  //the input command issued (same as text input box)
                this.disabled = true;  //can it be pressed?
                this.dReason = "This button is disabled for no reason.";
                this.visable = false;     //can it be seen?
                this.htmlID = htmlID;
                this.killMe = false;


        }

        Button.prototype.wake = function(that){
                if ( !that.killMe){
                        that.visable = true;
                        that.draw(that);
                }
```

```
}

Button.prototype.kill = function(that){
        that.disabled = true;
        that.visable = false;
        that.killMe = true;
        that.draw(that);
}

Button.prototype.setDisabled = function(that){
        that.disabled = true;
}

Button.prototype.setDisReason = function(that, s){
        that.dReason = s;
}

Button.prototype.debug = function(){
        console.log("[ btnInfo: name-" + this.label + " htmlID-" + this.htmlID + " command-"+ this.command +"]");
}

Button.prototype.clickMe = function(that){

                        if (this.disabled){
                                storyPrint(that.dReason);
                        }
                        else if (that.rTimer > 0){
                                this.refractMessage();
                        }
                        else {
                                var response = info.cBuildingID.getResponse(that.command, info.cBuildingID);
                                writeReponse (that.command, response);
                                that.rTimer = that.refract;
                        }
}

Button.prototype.create = function(that){
        var b = document.createElement("div");
        var c;
        //is disabled?
        if (that.disabled) {c = "dButton";} else {c = "aButton";}
        $(document).ready(function() {
                $("#menuTop").append("<p id= \"" + that.htmlID + "\" class=\"" + c + "\" >"+ that.label +"</p>");

                $("#"+that.htmlID).on("click", function(){
                        that.clickMe(that);
                        that.draw(that);
                        });

        });
}

Button.prototype.show = function(that){
		that.visable = true;
		//this.disabled = false;
	}

	Button.prototype.hide = function(that){
		//this.visable = true;
		that.visable = false;
	}

	Button.prototype.enable = function(that){
		//this.visable = true;
		that.disabled = false;
	}

	Button.prototype.draw = function(that){
		$(document).ready(function() {
			if (that.killMe){
				$("#"+ that.htmlID).toggleClass("aButton", false);
				$("#"+ that.htmlID).toggleClass("dButton", true);
				$("#"+ that.htmlID).fadeOut(2000);
			}
			else if ( !that.visable || info.cBuildingID  != that.locationID){
					$("#"+ that.htmlID).hide();
			}
			else{
					$("#"+ that.htmlID).show();
					if (that.disabled || that.rTimer > 0){
						$("#"+ that.htmlID).toggleClass("aButton", false);
						$("#"+ that.htmlID).toggleClass("dButton", true);
					}
					else
					{
						$("#"+ that.htmlID).toggleClass("dButton", false);
						$("#"+ that.htmlID).toggleClass("aButton", true);
					}
			}
		});
	}

	Button.prototype.splat = function(that, s){
		this.create(that);
		this.show(that);
		this.enable(that);
		this.draw(that);
		this.setDisReason(that, s);
	}

	Button.prototype.refractMessage = function(){
		storyPrint("you can \"" + this.command + "\" again in " + (this.rTimer/1000) + " seconds.");
	}

	Button.prototype.getID = function() {
			return this;
```

```
				}
```

Info.JS

```
	// JavaScript Document


	var info = {
		version: 0.09,
		state: "Alpha",
		author: "Ken Poirier",
		publisher: "Legenday Power Games",
		debug: true,
		verbosOn: false,
		country: "china",

		//unlocks
		canTravel: false, //change building
		canGoHome: false, //fast forward to next day
		canGetMail: false,
		//game stats
		storyArc: 0,
		teaDrank: 0,
		ppbClicks: 0, //how many times has the push paperwork button been pressed
		papersFiled: 0,

		//activity info
		cBuilding: "admin",
		//cBuildingID: china_admin.getID(), //old version
		cBuildingID: chinaComplex.bld_admin.getID(),
		cRoomID: chinaComplex.bld_admin.rm_darkRoom.getID(),
		lRoomID: chinaComplex.bld_admin.rm_darkRoom.getID(),
		hasTraveled: false,


	}

	var getLocID = function (bName, complex){
			if (bName = "admin") return complex.adminBld.getID();
	}

	var getScore = function(){
		return (info.ppbClicks * 10) + (info.storyArc * 100) + info.papersFiled - info.teaDrank;
	}
```

roomItem2.js

```
	// JavaScript Document
```

```
/// base class for ingame items and funiture



const Item = {
        disName: "cool item",
        altNames: ["none", "none"],
        desc: "This item does something cool.",
        canCarry: false,
        roomID: -1,
        hidden: true,
        inInv: false,
        validVerbs: [],

        //function
        getID: function() {
                return this;
        },

        isMyNoun: function(noun) {
                n = noun.toLowerCase();
                if (n == this.disName) return true;
                this.altNamesforEach(function(i){
                        if (i == n) {
                                return true;
                        }
                });
                return false;
        },

        getDisplayName: function() {
                return this.disName;
        },

        getDescription: function() {
                return this.desc;
        },

        takeAction: function(sVerb) {
                r = "";

                errorPrint("You can not \"" + sVerb + "\"  the \"" + this.disName + "\".");

                return r;
        },

        seeMe: function (){
        if (this.hidden) return "I don't see that right now.";
        else return this.desc;
        }

}
```

Start.JS

```
// JavaScript Document


//info

game_over = false;

//gameoptions
debugLog = true;

//Globals
country = "China";
possesive = "Chinese";
opponet = "Japan";
oPossesive = "Japanese";
nun = "none"; //special variable returns when parent isn't set


//the current building now in info.js
//cBuilding = "admin";
//cBuildingID = china_admin.getID();

//new clock
//gameClock = new clock();

//createInventory();

if (chinaComplex.bld_admin.rm_darkRoom.getVisits() == 0){
        //our first button
                var btnLamp = new Button("turn on light", 0, chinaComplex.bld_admin.rm_darkRoom.getID(), "get
lamp", "btnLamp");
                btnLamp.splat(btnLamp, "The light is already on.");
                var btnLookRoom = new Button("room", 0, chinaComplex.bld_admin.rm_darkRoom.getID(), "look",
"btnLookRoom");
                btnLookRoom.splatLook(btnLookRoom, "error");
                //btnLamp.splat(btnLamp, "The light is already on.");
                //if (info.debug) btnLamp.debug();
                //createInventory();

}


//Console layout
$(document).ready(function() {

        //version info
        $("#versionInfo").text("(version " + info.version + " " + info.state + ")");
```

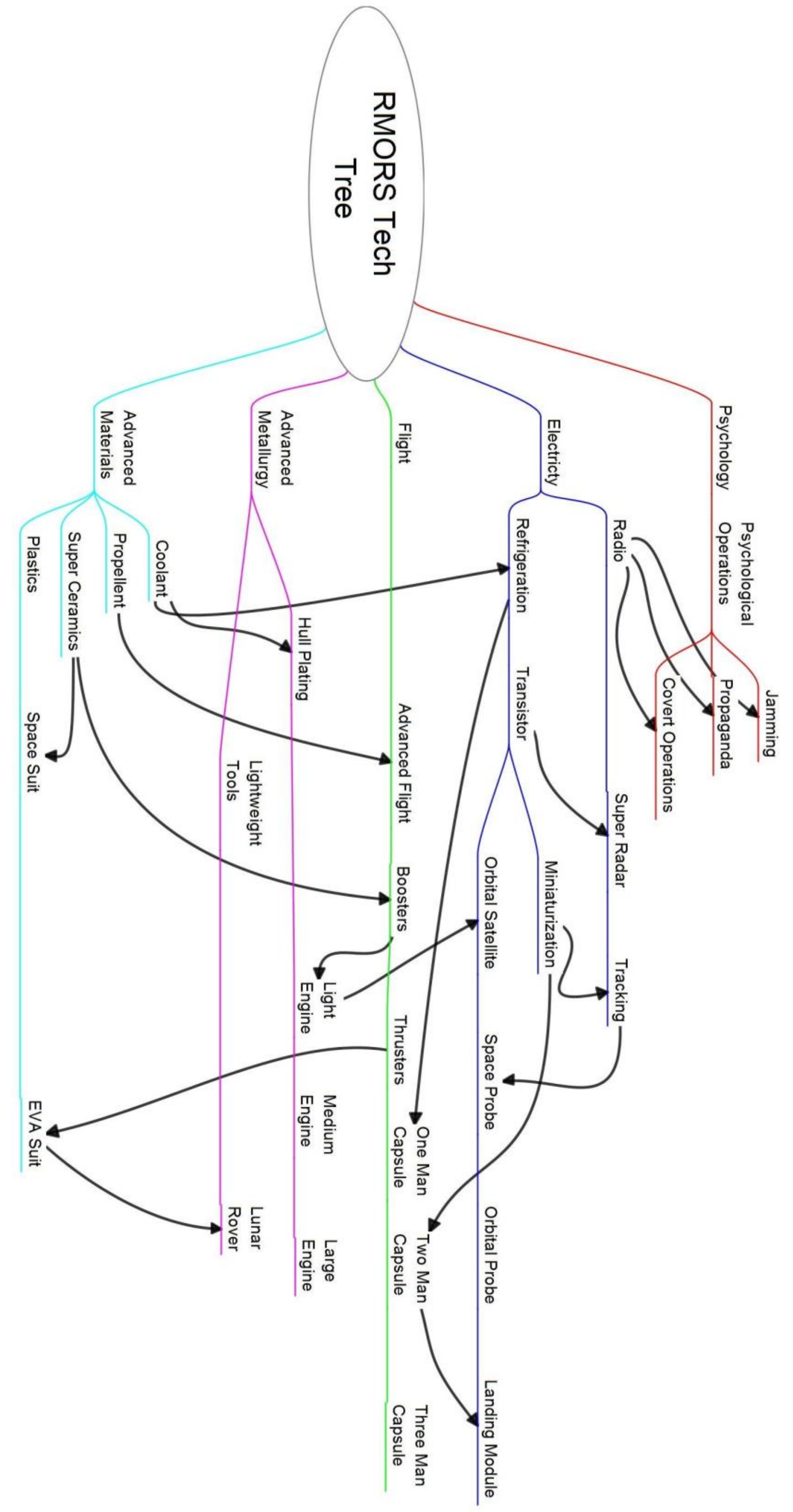
```
		//set date
		$("#time").text(clock.dateToSting() + " : Year of the " + getPet(clock.curDate.getFullYear()));

		//update date and time
		window.setInterval(function(){
			clock.updateClock();
			$("#time").text(clock.dateToSting() + " : Year of the " + getPet(clock.curDate.getFullYear()));
			$("#score").text(getScore());
		},100);


		//show console
		$("#console").fadeIn(3000);

		if (chinaComplex.bld_admin.rm_darkRoom.getVisits() == 0){

			//$("<p class=\"story\">" + china_admin.altLook + "</p>").hide().insertAfter("#storyLog").fadeIn(2000);
			$("#menuName").text(chinaComplex.bld_admin.rm_darkRoom.getDisplayName());
			//btnLamp.draw(btnLamp);
			$("<p class=\"story\">" + chinaComplex.bld_admin.rm_darkRoom.firstLook() +
	"</p>").hide().insertAfter("#storyLog").fadeIn(2000);
			chinaComplex.bld_admin.rm_darkRoom.addVisit();

		}

		//Input reader
		$("form").submit(function() {

			var input = $("#cInput").val().toLowerCase();
			var response = "none";

			writeReponse(input, response);

			$("#cInput").val("");
		});
	});
```

Appendix D – Strings

```
// JavaScript Document

var adminString = [];

adminString[0] = "The room is dark and cold.";

//the assistant arrives
adminString[1] = "The light from the electric lamp fills the room and shimmers against the glass windows. A small stack papers sit on your desk. The Chinese flag Chairman Moa personally gave you sit in it's box. You fix it to the pole in the corner and give it a salute. It is an honor to serve your countrymen. A dusty tea kettle sits top of an old wood stove.";
//assistant
adminString[2] = "Your assistant stumbles through the door and collapses in the chair at her desk. You remember what it is like to be young when getting up in the morning was your biggest challenge of the day.";
adminString[3] = "Your assistant shivers, and mumbles quietly. Her words are unintelligible. She heats up the tea kettel on the old wood stove.";
adminString[4] = "Your assistant sips her tea and stops shivering. her breathing calms as she exhales a cloud of warm breath in the cold morning air.";
adminString[5] = "Your assistant says she can help you. We cannot let Japan beat us to the moon.";

//Tea Status
adminString[6] = "You rub your eyes. You feel a bit tired. Some tea would be nice.";
adminString[7] = "You drink some tea.";
adminString[8] = "After some tea for yourself feel better and are ready to meet the challanges before you.";
adminString[9] = "You've had quite a bit of tea...";
adminString[28] = "Your assistant makes a wonderful cup of tea."
adminString[29] = "That\'s your assistant\'s job..."
adminString[30] = "The kettle is empty."
adminString[31] = "The tea is cold."
adminString[32] = "The tea is ready to drink."

//paperwork
adminString[10] = "This paperwork isn't going to push itself.";
adminString[11] = "It takes a lot of paperwork to build a great nation.";
adminString[12] = "Pushing paperwork makes the time fly.";
adminString[13] = "Your assistant helps you catch up on some of your paperwork.";
adminString[14] = "You decide that's it's best not to let these files fall into enemy hands. You shove the papers in the stove and shut the door.";
adminString[15] = "look at all these files to sort through. You are going to need more help.";
adminString[34] = "You\'re here to drink tea and do paperwork... and you\'re all out of papperwork."

//window looks
adminString[16] = "The sun has yet to rise. The sky is dark and full of stars. You look up at the moon. Your moon. China's moon. Just as the sun is racing to catch up with your day, so too will the Japanese be on your way to the moon."; //morning
adminString[17] = "The sun shines brightly over your little base. You wonder how long you can keep this place a secret."; //day
adminString[18] = "You look up and see the milky way crawl across the sky. The future of China is in the stars."; //night
adminString[27] = "The adminstraion building is nothing fancy. Some desks, lamps, cabinets, and a wood stove for warmth. None the less, it is heart of your operation here at the base."

//mail
```

```
adminString[19] = "You should check the mail...";
adminString[20] = "You check the mail.";
adminString[21] = "Some mail has been delivered. There is a stack of paper work to do as well as a check from the government. It seems you budget increase has been approved.";
adminString[22] = "Just some more paperwork.";
adminString[23] = "You recieved a check from the goverment.";
adminString[24] = "The mail contains a check and a stack of papers.";
adminString[25] = "You recieve a letter from the chairman. Apparently you are behind on your paperwork. Because of this, your funds are being witheld.";
adminString[26] = "Papers clutter the office. Perhaps you should attend to it."

//asistaint reccomends
adminString[33] = "Your assistant is a fine young example of Chinese womanhood. She makes a fine cup of tea and she is egar to please you."
```

Appendix E – Technology Tree


RMORS Tech Tree
Advanced Materials
Advanced Metallurgy
Flight
Electricity
Psychology
Coolant
Propellent
Super Ceramics
Plastics
Refrigeration
Radio
Psychological Operations
Hull Plating
Transistor
Covert Operations
Propaganda
Jamming
Space Suit
Lightweight Tools
Advanced Flight
Super Radar
Boosters
Orbital Satellite
Miniaturization
Tracking
Light Engine
Thrusters
Space Probe
Medium Engine
EVA Suit
One Man Capsule
Lunar Rover
Large Engine
Two Man Capsule
Orbital Probe
Landing Module
Three Man Capsule