The Endless Expanse Manual

The Endless Expanse

The Endless Expanse is a two dimensional role playing game that focuses on random content in an effort to create more replay value for the game. The game is for anyone who wants to play a game that is different in each playthrough. This manual will take you through how to play the game, but more importantly, how to customize your experience and add your own new and exciting content.

Playing the Game

The game is a role playing game that is meant to be played over and over. This allows you to create characters and worlds, and to continue making new ones if you are bored with the ones you have. The following sections take you through how to play the game as just a user, with no individual customization.

Creating a Game

The process of creating a game covers creating a world and then a character. Any character can play on any world: statistics and items transfer from world to world. The following images show the general process of creating a game.

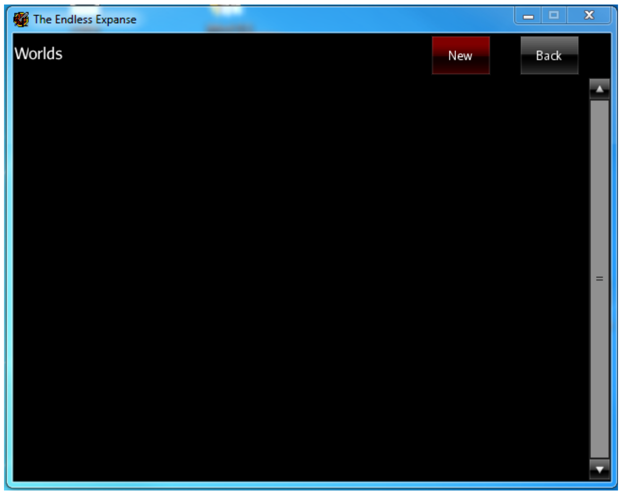


Fig 1: Blank World Screen

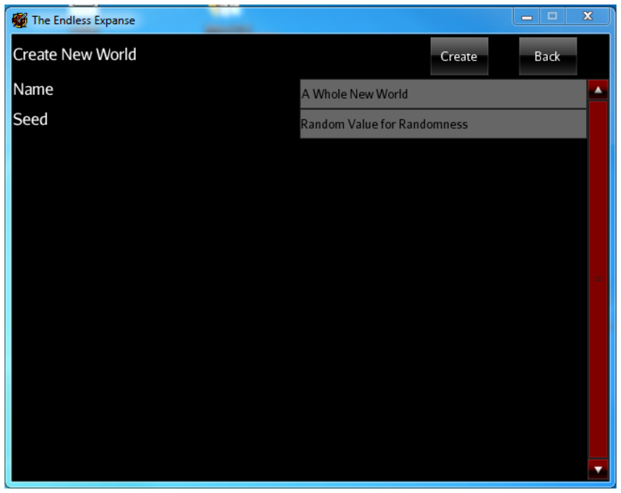


Fig 2: New World Screen

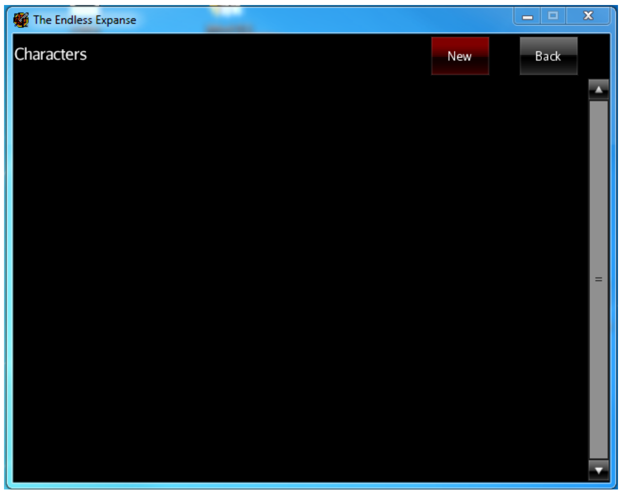


Fig 3: Blank Character Screen

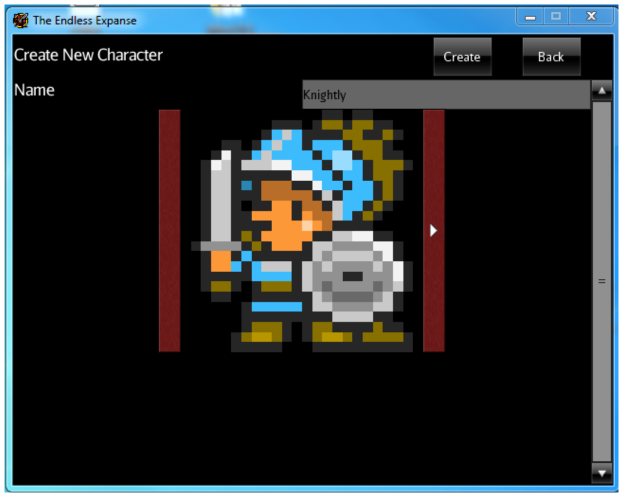
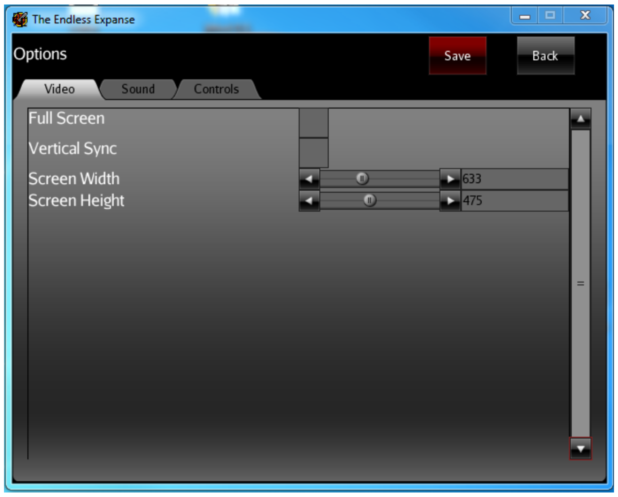


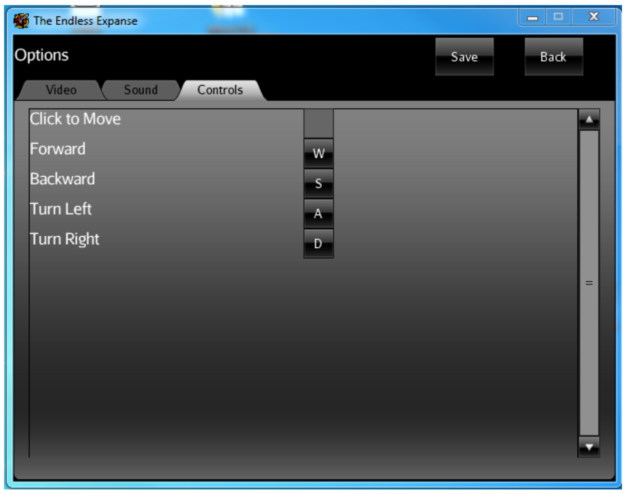
Fig 4: New Character Screen

Options

The game has limited options to choose from. There is room for expansion here, with some functionality currently. The interface has 3 tabs: Video, Sound, and Controls. Video has a few different options, such as resolution, full screen, and vertical sync. Sound currently has no options. Controls allows you to choose how to move around as well as if you want click to move on.



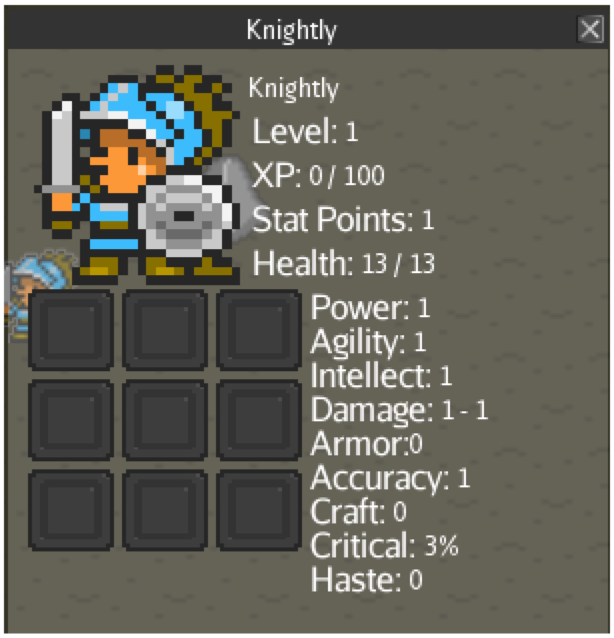
Full Screen
Vertical Sync
Screen Width 633
Screen Height 475



Click to Move
Forward W
Backward S
Turn Left A
Turn Right D

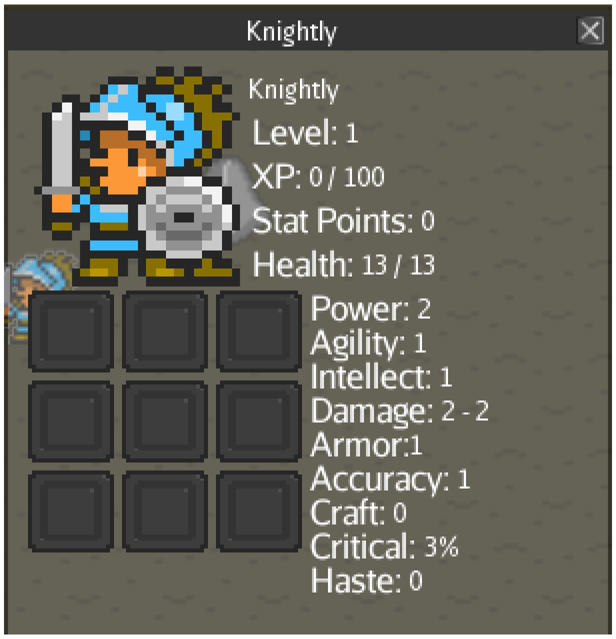
Statistics

Statistics are various attributes that a character has that determines that character's strengths and weaknesses. The stats come in a variety of forms, and can either be influenced by other stats or be standalone values. Equipment and Effects can modify stats. The three base stats can have points that are gained each level spent on them, in order to make them better.



Knightly

Knightly
Level: 1
XP: 0 / 100
Stat Points: 1
Health: 13 / 13
Power: 1
Agility: 1
Intellect: 1
Damage: 1 - 1
Armor:0
Accuracy: 1
Craft: 0
Critical: 3%
Haste: 0



Knightly

Knightly
Level: 1
XP: 0 / 100
Stat Points: 0
Health: 13 / 13
Power: 2
Agility: 1
Intellect: 1
Damage: 2 - 2
Armor:1
Accuracy: 1
Craft: 0
Critical: 3%
Haste: 0

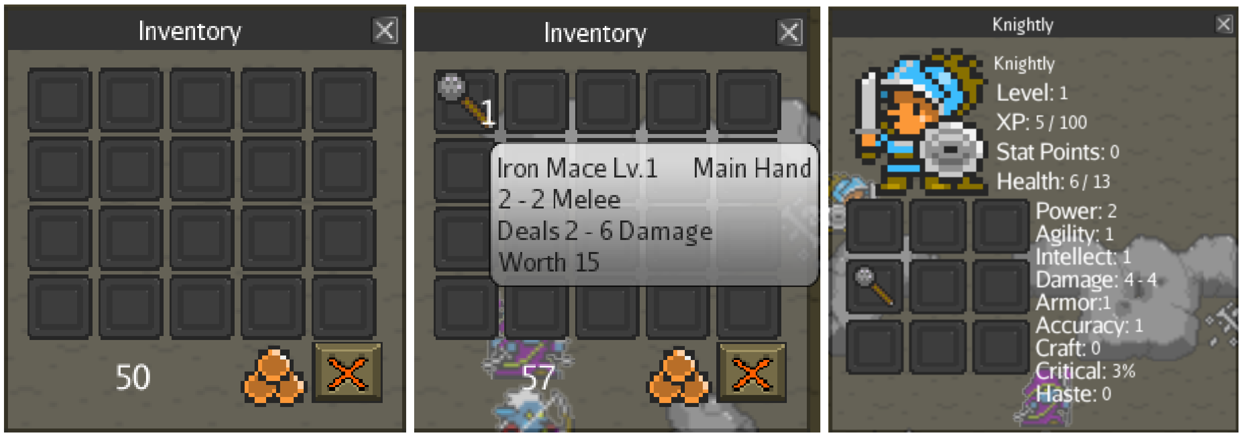
- Level: Standalone stat that is a general teller of a character's abilities. Increases when XP reaches 100.
- XP: Shows how much a character has done this level. After 100, resets to 0 and increases Level (which also increases Stat Points).
- Stat Points: Remaining points to spend on Power, Agility, or Intellect.
- Health: This stat depends upon Level, and when it reaches 0, the character is destroyed (temporarily).

- Power: A stat that represents physical strength. Increases Melee weapon damage and Armor.
- Agility: A stat that represents dextrous prowess. Increases Ranged weapon damage and Accuracy.
- Intellect: A stat that represents knowledge and wisdom. Increases Magik weapon damage and Craft.
- Damage: Damage that is dealt with the current weapon. No weapon is Melee.
- Armor: Decrease damage taken by this value, down to a minimum of 0.
- Accuracy: This value, which is modified by Level and Agility, determines how easy it is to hit something. You should have about a 50/50 chance to hit something whose Level is the same as the value of Accuracy.
- Craft: Increases damage and healing of any item used.
- Critical: Percent chance to deal double damage with an attack.
- Haste: How often you can act. You can do one action even 1 - 0.01*Haste seconds. Maxes out at 90, where you can act 10 times per second.

Items and Inventory

Items are objects that the player collects throughout play. These items can have just a use, where they are able to be used in order to grant an immediate effect. Items can also be equipped in certain equipment slots. Items can be bought and sold with certain npcs by clicking on them. Items will tell you what they do by simply scrolling over them.

The Inventory is where you store your items, as well as currency. Currency is exchanged with npcs for items, and you can sell your items to get more currency. Currency can also be spent to fast travel to places on the map, and the Exploration section will cover.

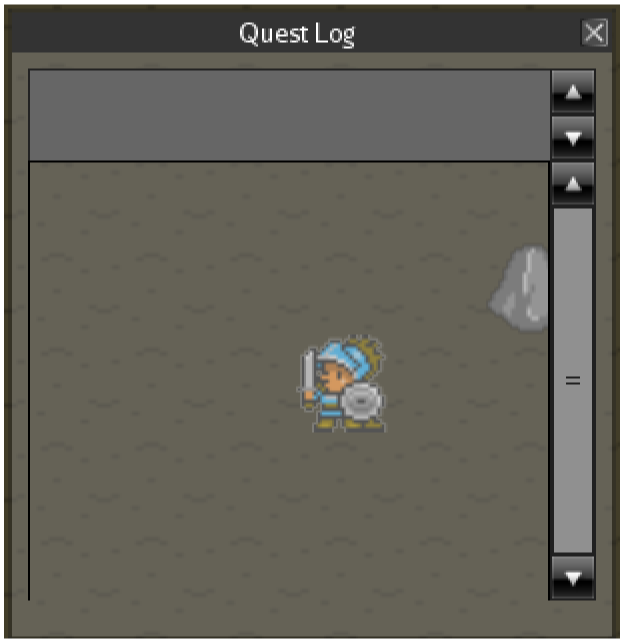


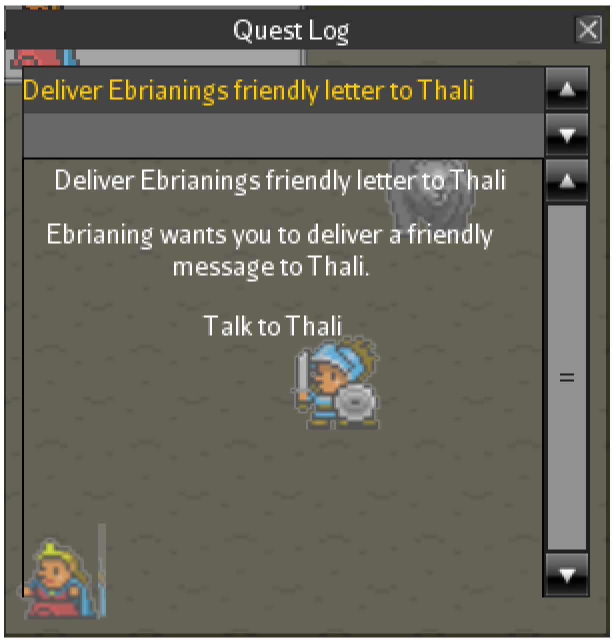
Iron Mace Lv.1 Main Hand
2 - 2 Melee
Deals 2 - 6 Damage
Worth 15

Knightly
Level: 1
XP: 5 / 100
Stat Points: 0
Health: 6 / 13
Power: 2
Agility: 1
Intellect: 1
Damage: 4 - 4
Armor: 1
Accuracy: 1
Craft: 0
Critical: 3%
Haste: 0

Quests

Quests are objectives that are given to the player via NPCs and are used to motivate the player. Quests grant rewards afterwards that will grant the player some items, currency, and XP. Quest requirements can range from talking to other NPCs, to gathering items, to killing monsters. Quests can be set up to have some historical event required, although there are none in the base game that do this as of now.





Deliver Ebrianings friendly letter to Thali

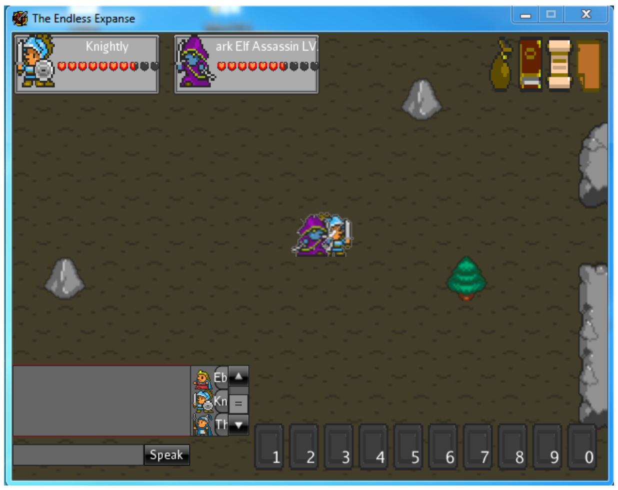
Deliver Ebrianings friendly letter to Thali

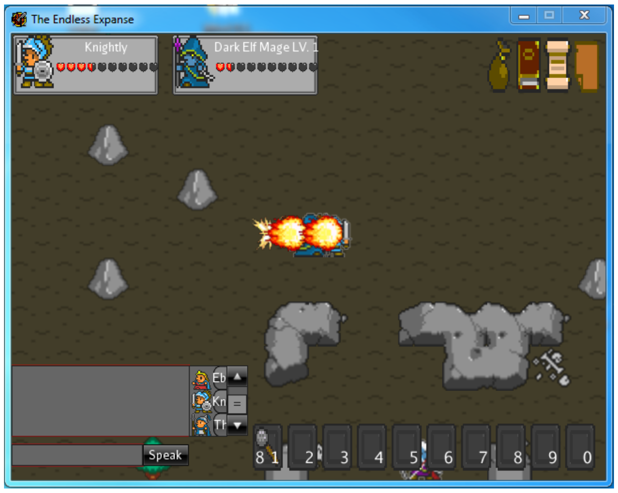
Ebrianing wants you to deliver a friendly message to Thali.

Talk to Thali

Combat

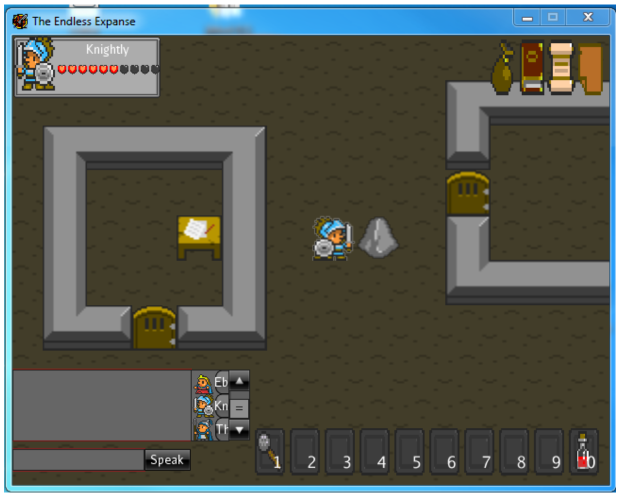
Combat is the exchange of blows between player and monsters. This is in the form of attacks and items. If an item is used, then we see a cooldown for the item in the action bar. Every use and attack have different visual and sound effects, livening up the game and making combat more exciting.

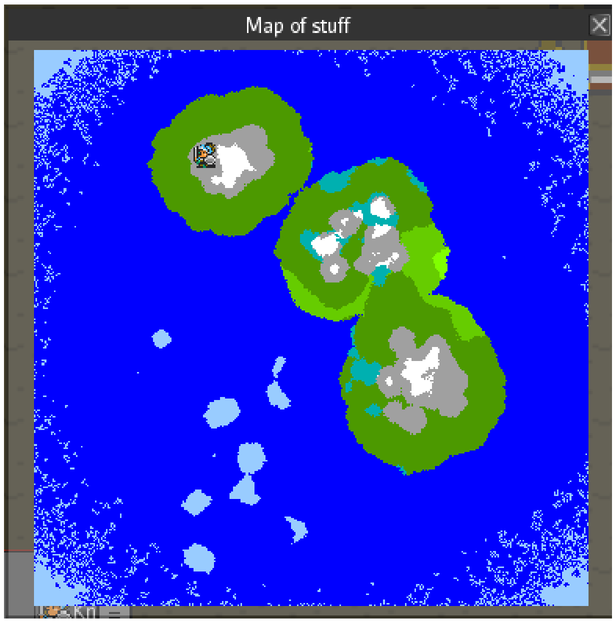


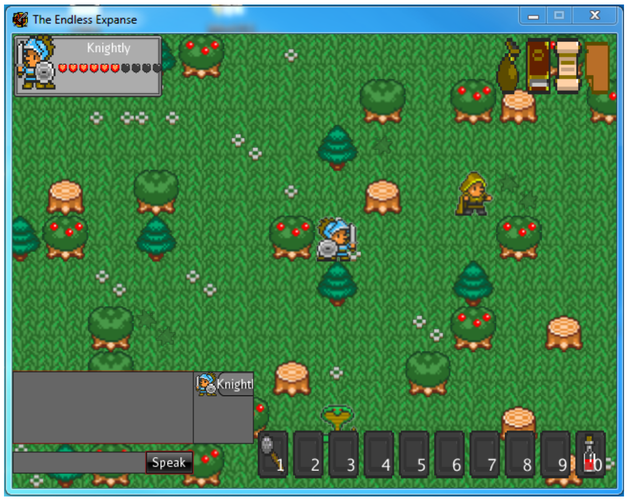


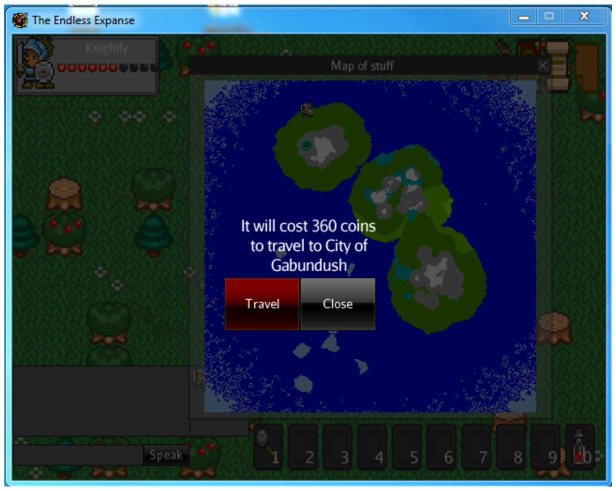
Exploration

Due to the random content of worlds, The Endless Expanse is all about exploring. Walk around the world to explore new areas, and open up the map in order to look around you. If you see any spot on the map that isn't your character, it is someplace you can fast travel to, for a price. The price is based upon how far away you are. You can add your own versions of these to the world folder: look for the tags folder inside the world folder in order to add your own. Follow the basic outline listed there, where the numbers are pixel coordinates and the words are the displayed name.









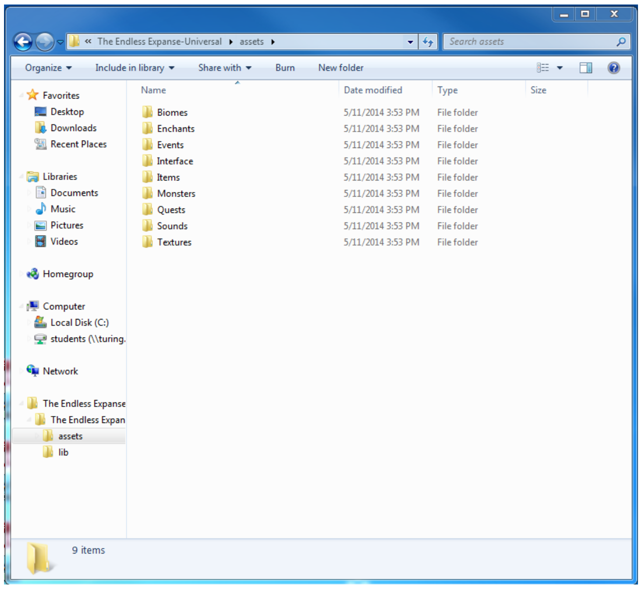
It will cost 360 coins to travel to City of Gabundush

Customizing the Random

One of the major features of the game is the idea of being able to customize the randomness of the game. This ranges from biome modification, image changes, items, event types, and quest types. This section will give you a brief overview of how to customize your experience.

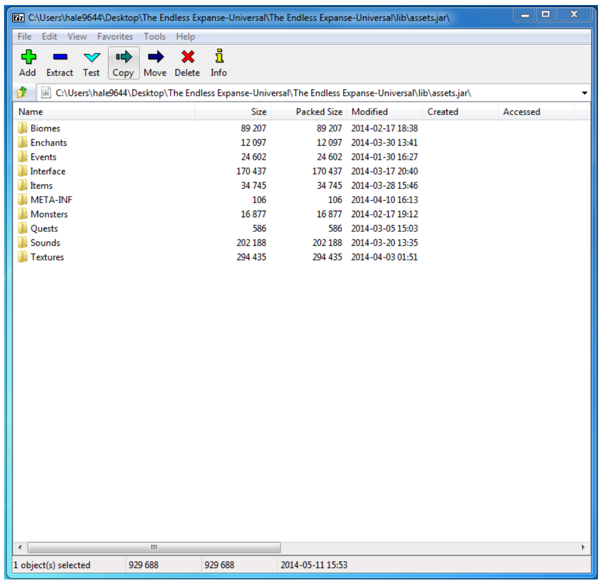
Changing Assets

Due to the current library, assets are split into two sections: assets.jar and assets folder. The folder is where you change biomes, enchantments, events, items, monsters, and quests, as well as localization for any of these things. The jar file is where you change any images or sounds, including in the interface. It is currently important that the knight, mage, priest, rogue, and archer entities remain with their EXACT names. Open the Jar file with any zip file opener, such as winRar or 7zip.



Name	Date modified	Type	Size
Biomes	5/11/2014 3:53 PM	File folder	
Enchants	5/11/2014 3:53 PM	File folder	
Events	5/11/2014 3:53 PM	File folder	
Interface	5/11/2014 3:53 PM	File folder	
Items	5/11/2014 3:53 PM	File folder	
Monsters	5/11/2014 3:53 PM	File folder	
Quests	5/11/2014 3:53 PM	File folder	
Sounds	5/11/2014 3:53 PM	File folder	
Textures	5/11/2014 3:53 PM	File folder	

9 items



C:\Users\hale9644\Desktop\The Endless Expanse-Universal\The Endless Expanse-Universal\lib\assets.jar\

Name	Size	Packed Size	Modified	Created	Accessed
Biomes	89 207	89 207	2014-02-17 18:38		
Enchants	12 097	12 097	2014-03-30 13:41		
Events	24 602	24 602	2014-01-30 16:27		
Interface	170 437	170 437	2014-03-17 20:40		
Items	34 745	34 745	2014-03-28 15:46		
META-INF	106	106	2014-04-10 16:13		
Monsters	16 877	16 877	2014-02-17 19:12		
Quests	586	586	2014-03-05 15:03		
Sounds	202 188	202 188	2014-03-20 13:35		
Textures	294 435	294 435	2014-04-03 01:51		

1 object(s) selected 929 688 929 688 2014-05-11 15:53

Items

Items are in JSON format in the Items folder of the assets folder. The JSON is in the following format, with descriptions of each field below:

```
{
	"names":[],
	"enchants":[],
	"onUse":[],
	"equipLocations":[],
	"onEquip":{},
	"weaponType":"",
	"levelGap":,
	"rarity":,
	"availableAt":,
	"image":"",
	"attackAnimation":{}
}
```

- names: list of strings that the item can be named. These are passed through the localizer.
- enchants: list of enchantment ids that this item can have on it.
- onUse: list of use cases that this item can have
- equipLocations: list of places this item can be equipped.
- onEquip: damage set of what happens when this item is equipped.
- weaponType: string representing what type of weapon this is for damage determination.
- levelGap: how many levels to skip before getting an upgraded version of the item.
- rarity: how rare this item is. Lower numbers are rarer.
- availableAt: the first level this item can be found or purchased at.

- image: the image location to use for this item.
- attackAnimation: use animation to use while attacking with this item.

Enchants are in the enchant folder and look like:

```
{
	"id":"",
	"rarity":,
	"availableAt":,
	"attackTarget":"",
	"useTarget":"",
	"onUse":{},
	"useAnimation":{},
	"onEquip":{},
	"attackAnimation":{}
}
```

- id: string id of the enchantment.
- rarity: how rare the enchantment is. Lower numbers are rarer.
- availableAt: what level this enchantment is available at.
- attackTarget: who is targeted by this enchantment when attacking.
- useTarget: who is targeted by this enchantment when using the item.
- onUse: use case when using the item.
- useAnimation: animation when using the item.
- onEquip: how the item modifies the user while equipped.
- attackAnimation: animation when attacking with the item.

Look at the various already existing items and enchantments to get a better view of how to create your own.

Monsters

Monsters, like Items, are also in JSON format, but with different key-value pairs, as shown below:

```
{
	"name":"",
	"images":[],
	"initialSprite":"",
	"speed":,
	powerBase:,
	agilityBase:,
	intellectBase:,
	possibleBiomes:[],
	"ai":[],
	"defaultAttack":"",
	"attackAnimation":{}
}
```

- name: localized name

- images: array of image sets. These sets are animation loops for various actions, and only left and right are needed. The initial array is for multiple animation sets for the same monster to choose from at creation time.
- initialSprite: the string id of the initial animation to use. Usually ‘left’.
- speed: the animation speed (in seconds) for the monster.
- powerBase: how many points of power per level the monster has
- agilityBase: how many points of agility per level the monster has
- intellectBase: how many points of intellect per level the monster has
- possibleBiomes: array of biome names the monster can appear in.
- ai: array of ai to assign to this monster.
- defaultAttack: Which base stat to use for attacking with.
- attackAnimation: animation to use while attacking.

Biomes

Biomes use JSON as well, and biomes are generated at the time you create the world. The represent each pixel on the map, and each pixel represents 64x64 tiles in game.

```
{
	"name":"",
	"color":"",
	"habituality":,
	"temperature":[],
	"precipitation":[],
	"elevation":[],
	"generation":{}
}
```

- name: name of the biome for use with monster creation.
- color: hexadecimal color code for the pixel on the map.
- habituality: how likely a town can be created on the biome. At 0, there is no chance.
- temperature: the temperature range this biome can be created in.
- precipitation: the precipitation range this biome can be created in.
- elevation: the elevation range this biome can be created in.
- generation: complex json that is used to generate the 64x64 tiles. This contains the locations of all images used in the area, what is laid down as a base, and what kinds of structures can spawn in the biome.

Event Types

Event types are used to determine what kinds of events will happen during history generation. These are things like a faction being created or an npc finding a dog. The Global folder is for events that are relevant to the entire world, and the Local folder is for events that are just required for specific areas. This is, of course, also in JSON:

```
{
	"id":"",
	"variables":{},
        "counts":{},
        "changes":{},
        "causes":{},
        "dependencies":{},
        "lockouts":{}
}
```

- id: id of the event
- variables: map of name to type of variables for the event, like factionName.
- counts: structure that allows this event to count as some other event. Like a faction splitting off from another faction also counts as a faction creation for the one that split off.
- changes: these are parts in game that need to be changed as the event happens.
- causes: these are events that can be caused by this event in the future. A hero must die of old age if she doesn't die of something else first.
- dependencies: other events with variables that match with some variables of this event that must have happened before this event can happen. A faction can't fall if it hasn't been created.
- lockouts: other events that must match, like dependencies, to specific variables that if any of the lockouts have happened, this event can no longer be used as a dependency. A faction can't go to war if it has fallen.

Quest Types

Quests use a framework that allows us to have general or specific requirements for the player and the npc giving the quest. We can do this with the following JSON format:

```
{
        "id":"",
        "kill":{},
        "item":{},
        "talk":{},
        "jobs":[],
        "biomes":[],
        "history":[],
        "chain":[],
        "offerRate":,
        "talkCount":
}
```

- id: id of the quest.
- kill: map of monsters the player must kill as a requirement. any means any monster
- item: map of items the player must collect as a requirement. any means any item. For this case, only stackable items (those that have no enchants and the same effect every time) will be used.
- talk: map of npc jobs that can be talked to from certain jobs.
- jobs: array of npc jobs that can offer the quest

- biomes: array of biomes the npc must be in to offer the quest
- history: array of historical events that must have happened to the npc in order to offer the quest.
- chain: array of quest templates the player must have completed for this npc previously in order to take this quest.
- offerRate: how many times the player can be offered the quest. 1 is the usual case, but can be more than 1, or infinitely repeatable by making this less than 1.
- talkCount: how many different npcs need to be talked to for the talking requirement of the quest.

About the Author

Christopher Hale is a Computer Scientist who has recently graduated from Pacific University. He decided to try a game for the challenge that it presented with visible results.

Glossary

AI: Artificial Intelligence. Meaning actions that some computer controlled system takes based upon some set of input.
Character: A person in game, usually referring to the player.
Click to Move: Whether or not a left mouse click on a location should attempt to move your character to that location.
Customization: Content that is changed by the user to fit their individualized experience.
Effect: Structure that runs for some amount of time. During this time, damage is dealt or healed, and stats can be affected either negatively or positively.
Equipment: An item that can be equipped to the character, granting bonuses while it remains equipped.
Full Screen: Whether the game should be played in full screen mode or not.
JSON: JavaScript Object Notation is a key-value pair system that allows us to store data is a flexible way that is readable by both humans and computers.
Localization: Using an arbitrary string format in order to translate printable strings into other languages. These generally have english descriptors separated by periods.
NPC: Non-player Character. Referring to creatures in game that do not attack the player.
Random Content: Content for some product that is created at the time the product is used.
Replay Value: The chance that a game will be played more than once by a player.

Resolution: The screen width and height.
Role Playing Game: A game that uses an increase of power with players as a motivation. Usually has a driving story that acts as a finishing motivation for players.
Stats: Short term for Statistics. Statistics are attributes that determine a character's strengths and weaknesses.
Two Dimensional: Some structure that uses only two of the three dimensions: width, height and depth. Usually the structure recognizes that there is a third unused dimension.
Vertical Sync: Whether to limit the game to 60 frames per second. This could help avoid some image tears and artefacts.
World: An in game world that the player will play the game on.
ZIP File Opener: A program that can open compressed files. They have to at least be able to open .zip files, although they can open other types of compressed files as well.

Index

Random Content	1, 5, 6
Replay Value	1
Resolution	2
Role Playing Game	1
Sound	2, 5, 6
Sprite	8, 9
Statistics	1, 3, 4, 9
Temperature	9
Tile	9
Two Dimensional	1
Use(Item)	4, 5, 7, 8
Vertical Sync	2
Weapon	4, 7
World	1, 2, 5, 9
XP	3, 4
ZIP	6