# IT 140 Design Document Template

## Instructions

Fill out the sections below. Be sure to remove the bracketed text before submitting your Design Document.

Tyler Anderson

## Storyboard (Description and Map)

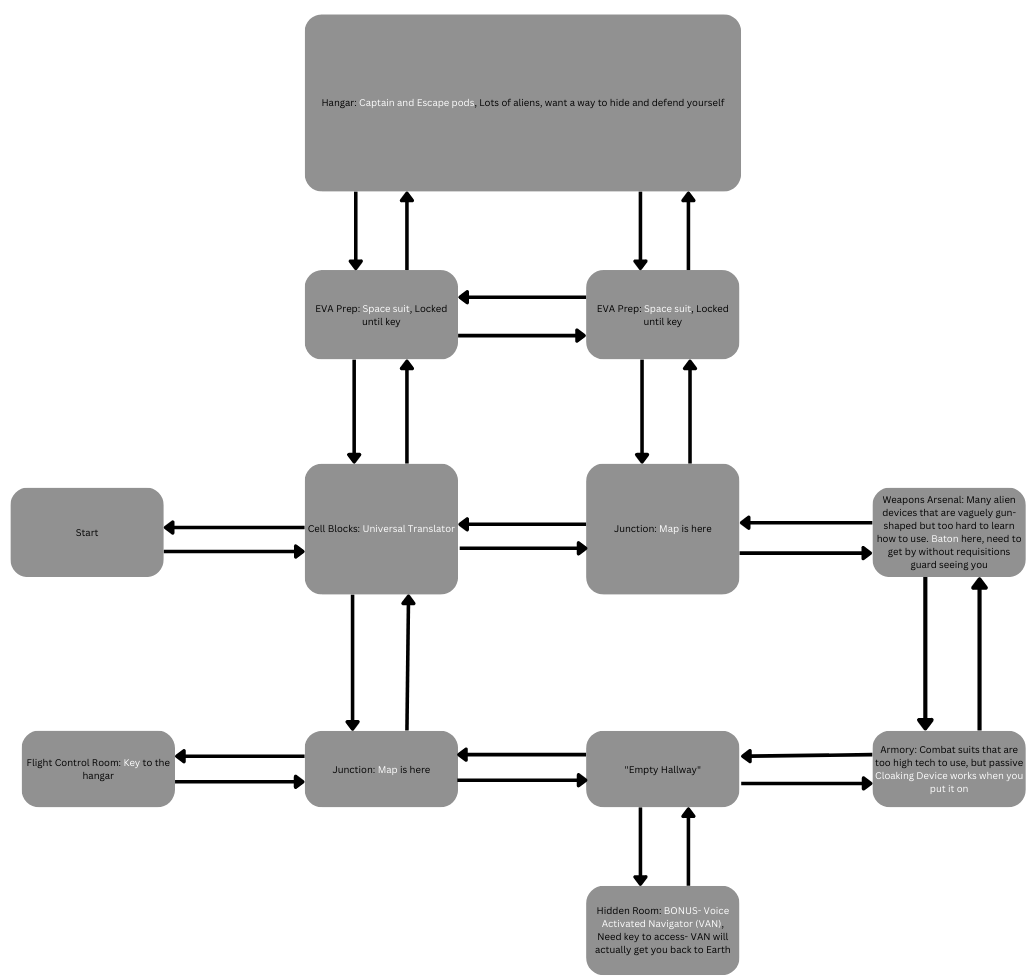
Theme and Villain:

Set on an alien spaceship, this is going to be a text-based game of harrowing suspense as you have been abducted by space aliens! You suffered the strange, cruel interrogations and experiments for who knows how long, all of which were led by a callous and calculating higher alien you have come to know only as The Captain. But, they finally made a mistake. You were able to free yourself from your shackles and roam the ship! Your first taste of freedom…

Storyline with Rooms and Items:

You need to sneak to an escape pod and escape back to Earth. There are 12 total rooms, and scattered throughout are the objects you will need to escape. The items are the Universal Translator, a Datapad containing a Map, a well-weighted Baton made of some pearlescent metal you cannot understand the makeup of, a Cloaking Device to hide yourself, a Space Suit to survive the alien environment in case something happens to your pod on the return trip, and finally A Key to the Escape Pods. With adequate development time there may be a hidden bonus item as well: The VAN- short for Voice Activated Navigator, which will ensure you get home to Earth without relying on guesswork and luck…

**Map:**



## Implementation:

## Pseudocode to “Move Between Rooms”

Goal: Allow player to move between rooms. This will require the game to accept player input via a prompt, to know what the current room is, what directions are valid for the player to move in based on those directions, and what the next room is based what is connect to the current room.

Potential issues:

* Invalid input from player, i.e. player didn’t choose move or grab.
* Invalid direction, i.e. room the player is in does not have that direction as a possibility, there is no room connected to this one that way
* Game needs to be aware of a set of information about the room

Pseudocode:

*Needed elements:*

* Main() function which will contain the loops for getting the player’s choices as input, decide which function is needed, and then to pass the choice to the relevant function for processing. In this case main() needs to pass to the move\_room() function as an argument to the move\_choice parameter.
* Namedtuple containing all information about a room, accessible by RoomName.RoomAttribute. Namedtuples will be global.
* Possible directions of a room, stored as a list within the room’s namedtuple, so that way it can be looped through without needing to hardcode each direction for each room. This will be accessible by RoomName.possible\_directions.
* Which rooms are connected to which other rooms and by what direction. Will be contained as RoomName.possible\_directions. RoomName.possible\_directions will be a dictionary with the direction the next room is in as the key, and the room’s name as a string for that dictionary’s value.
* A dictionary with the room names as strings which will be used to map the string versions of the names back to the tuple versions of the new room, stored globally as rooms\_dict.
* Variable holding the current room the player is in, stored in current\_room and updated by the move\_room() function.

*Input:*

* String containing the words “go [DIRECTION]”, stored in the variable named ‘choice’. ‘choice’ will be passed as the argument to the move\_room() function’s parameter, move\_choice.

**Processing on next page.**

*Processing:*

1. def move\_room(move\_choice): | taking the move choice input as an argument
2. for paths in current\_room.possible\_directions: | Looping through the possible directions defined in the current\_room’s namedtuple for a potential match to move\_choice
3. if move\_choice == paths: | checking that the direction the player chose to move towards was valid by looking through a list of valid directions contained in a larger namedtuple.
4. Set current\_room = new room in that direction | will need to inform computer what room is in that direction, accessed by comparing the namedtuple direction:room-string pairs to the room-string:room-tuple pairs in the rooms\_dict.
5. Print feedback to player, such as ‘You move towards {New Room’s Name}.’
6. return current\_room | returns current\_room so that the information is passed to main(), informing the room should be changed. NOTE: will need this set in an assignment statement in main() to actually change the room, such as current\_room = move\_room(direction).
7. else: | direction chosen was invalid
8. Print feedback to player, such as ‘You see no way to go there.’
9. Return None | Will return None to inform main() that this was not a valid choice and to repeat the input loop asking for another player choice.
10. Stop this function, and return to greater game loop for next input.

## Pseudocode to “Get an Item”

Goal: Allow player to grab an item in the room they are in. This will require the game to accept player input via a prompt, to know what the current room is, what item is in that room, determine if that input was valid based on what item is in the room, and put valid items into a player held inventory, which will be updated as the game progresses.

Potential issues:

* Invalid input from player, i.e. player didn’t choose move or grab.
* Invalid item grab, i.e. that item doesn’t exist in this room.
* Game needs to be aware of a set of information about the room, in this case: what item is in the room.
* Player picks up multiples of the same item, causing inventory clutter with duplicates and breaking immersion.

Pseudocode:

*Needed elements:*

* Main() function which will contain the loops for getting the player’s choices as input, and pass the choice to the relevant function for processing. In this case choice is passed to the grab\_item() function for the item\_choice parameter.
* Namedtuple containing all information about a room, accessible by RoomName.RoomAttribute.
* What item is in this room, accessible by RoomName.room\_item
* List containing all items the player has collected so far (starts empty), stored in player\_inventory.
* Variable holding the current room the player is in, stored in current\_room and updated by the move\_room() function above.

*Input:*

* String containing the words “grab [ITEM]”, stored in the variable named ‘choice’. This will be passed as argument to the grab\_item() function

**Processing on next page.**

*Processing:*

1. def grab\_item(item\_choice): | taking the item choice input as an argument
2. if item\_choice == current\_room.room\_item: | checking that the item the player would like to pick up is in this room, accessed from within a larger namedtuple.
3. if item\_choice in player\_inventory: | avoiding the player picking up multiples of the same item
4. Print feedback to player such as ‘You already have the {item\_choice}’
5. Return None | inform main() that input was invalid, nothing has changed and to repeat the input process
6. Else: | Item exists in this room and the player does not have it yet
7. Player\_inventory.append(item\_choice) | add item to player inventory
8. Print feedback to player, such as ‘You pick up the {item\_choice}.’
9. return player\_inventory | passes the resulting list with the new item appended to it back up to main()
10. else: | The item the player is looking for is not in this room
11. Print feedback to player, such as “You don’t see that item anywhere in here”
12. Return None | inform main() that nothing new has changed and to repeat the input process
13. Stop this function, and return to greater game loop for next input.