LAB 7

Objectives

1. Identify the entities of a detailed design for your project.
2. Create detailed designs for a subset of the entities of your project.
3. Assess your team's technical capability compared to the technical needs of the project.

Designers need to specify the details of the entities that make up the system. These definitions should be sufficiently detailed that the design can be given to a developer and the developer can create the entity as envisioned by the designer.

Once your team starts to develop a design, you should also be developing a better understanding of the technologies and skill levels needed to build the product. As a separate task, this lab will also provide a chance for you to assess your team's capability to work on the project and identify learning or skill development you may need.

Procedure

Step 1 – Draft a list of entities for your project

You should consider the following types of entities:

- Screens (or Web pages)
- Database tables
- Files (e.g., data that is stored as part of the system but not stored in a database)
- Code (modules, objects, or functions)

Use Figure 7-1 to list all the system entities that you can identify. A good way to start is to pick one area and focus on that. For example, if your system has a significant user interface, start by trying to name all the screens that would comprise your interface. For each entity you list:

- Enter a type, e.g., "screen"
- Give it a meaningful name, e.g., "CustomerProfile"
- Provide any short notes or explanation needed to identify the screen, e.g., "This screen captures customer information and preferences."

Step 2 – Create detailed designs for at least 4 of your entities.

You will not be able to design all the entities of your system in this lab, but this step will get you started. Pick 4 entities that you think you understand the best at this point, and create a design for them. Every entity should have a name, type, and design details. Templates are provided to help you create detailed design for screens, database tables, and code functions.

Step 3 – Review your detailed designs.

After creating your designs, review them for completeness and clarity. Ask yourself this question: "If I was the developer and a designer handed me this design, would I know what to build without needing to ask a lot of questions?

If you have created the design entities as a team, set them aside for a few minutes before review each one. If you have worked in sub-groups within your team to create the designs, then exchange designs so the reviewer is a different person than the creator of a design.

Revise your designs based on the review.

Step 3 – Assess your team's capability to complete this project.
Once you have an architectural overview and the beginning of a design, you should be able to assess capability and identify things that someone on the team may need to learn. Use Figure 7-5 to summarize this information.

3.A – List the technologies you need for your project using the column on the left. Consider things such as programming languages, operating systems, specialized data sources, software libraries, support tools, and hardware.
3.B – List each team member at the top of a column, and then evaluate that person's knowledge of the technology in each row. For the column for each team member, use the following values:

1 – No knowledge or not much relative to the needs of this project
2 – Enough knowledge to accomplish part but not all of this project
3 – Knowledge probably sufficient for this project

3.C – Discuss within your team how you will start to gain capabilities that you are missing. You do not need to turn in results of this discussion in this lab, but will need to address this in the coming weeks.

What to Turn In

In order to obtain full credit for this lab, ***each team*** must turn in:

1. Figure 7-1 – Possible System Entities
2. Detailed designs for at least 4 entities in your system. Use the templates in Figures 7-2 through 7-4 to get started.
3. Figure 7-5 – Team Capability Assessment

Figure 7-1 – Possible System Entities

Product: Cow Panic!
Team: 01
Date: 3/30/2016

Type	Name	Description or Notes
Screen	MainMenu	Presents the user with 3 menu options: **Play**, **Settings**, **Extras**, and **Quit**(?)
Screen	Title Screen	A title splash screen that presents the game name and logo (if we have one) and prompts the user to tap the screen to continue to the **Main Menu**
Code (module)	modeEndless	The endless game mode, in which cows spawn and move across the screen endlessly until the player misses a cow, after which the game will end and the player will be sent to the **Scoreboard**
~~***Code (module)***~~	~~***modeCampaign***~~	~~***The campaign game mode, which is a level-based game mode. Cows spawn for a set period of time, and as the game goes on they will spawn more frequently and move faster. If the user successfully abducts all the cows, he/she will be presented with a "you won?" screen and taken to the Store screen***~~
~~***Screen***~~	~~***Play***~~	~~***Presents the user with 2 menu options: Endless and Campaign***~~
Screen	Settings	Presents the user with a number of game settings that can be modified, such as volume sliders (game music/sound effects) and an option to enable/disable cow puns (this is critical)
Screen	Extras	Presents the user with two options, the **Scoreboard** and the **Credits**
Screen	screenCredits	Presents the user with the game credits, either scrolling down the screen or centered in the middle of the screen (maybe in a fancy box)
Screen	screenScoreboard	Presents the user with the **Scoreboard**, the high scores from the **Scoreboard** database in the form of a chart (player initials followed by score, such as DMG - 1365)
Screen	In-Game	The screen upon which the game will be played. Cows are moving across the screen and you use your UFO to abduct them.
~~***Screen***~~	~~***screenStore***~~	~~***Presents the user with a store interface, in which he/she can spend coins on UFO upgrades, which will make their UFO work better (faster abduction, faster movement, etc.)***~~
~~***Code (function)***~~	~~***functionStore***~~	~~***Function that allows the user to buy UFO upgrades. If the user requests to buy a UFO upgrade, it will check to see if he/she has enough coins, and if he does it will add that upgrade to the UFO.***~~
Files	fileCredits	Text file containing the game credits, all of our names and our roles in development.

Code (function)	Quit	Main menu option that, when pressed, will present the user with a menu box asking if he/she really wants to quit (yes/no). Upon pressing yes, the game will close, and upon pressing no, the game will return the user to the main menu.
Database table	databaseScoreboard	A local database table that stores the scores achieved in the **Endless** game mode and saves the highest 5 scores
~~***Files***~~	~~***GameStats***~~	~~***Game file that keeps track of how many coins the player has and what upgrades their UFO has.***~~
Code (function)	***checkCollision***	***A function for checking collisions between two objects; it accepts a LayerMask as an argument to only check for ojects on a certain layer***
Code (script)	***Moove***	***This is the function that handles cow movement by having them move to the right as long as they have not collided with the tractor beam***
Code (script)	***abduct***	***This spawns a tractor beam object that then extends fully, stays out a few seconds, and then retracts***

Figure 7-2 – Template for Detailed Design for a Screen

Name: Title

Type: Screen

Purpose: This screen is needed to meet requirement: (We did not have a requirement for this; we decided to include a title screen later on.)

Description: Figure 1 shows the layout for this screen. This screen introduces the user to the game.

The screen contains the following elements: -Game Logo, the game's "trademark".

Layout:

Figure 1 - Title Screen

CI102

Name: Main Menu

Type: Screen

Purpose: This screen is needed to meet requirement: menuMain.

Description: Figure 2 shows the layout for this screen. This screen contains the main menu, which allows users to navigate the various other screens within the game, including the Play menu and the Settings menu.

The screen contains the following elements: - Boxes which, when swiped upward on by the user, cause an option to be selected and take the user to its corresponding menu.

Layout:

Figure 2 – Main Menu Screen

CI102

Name: Play Menu

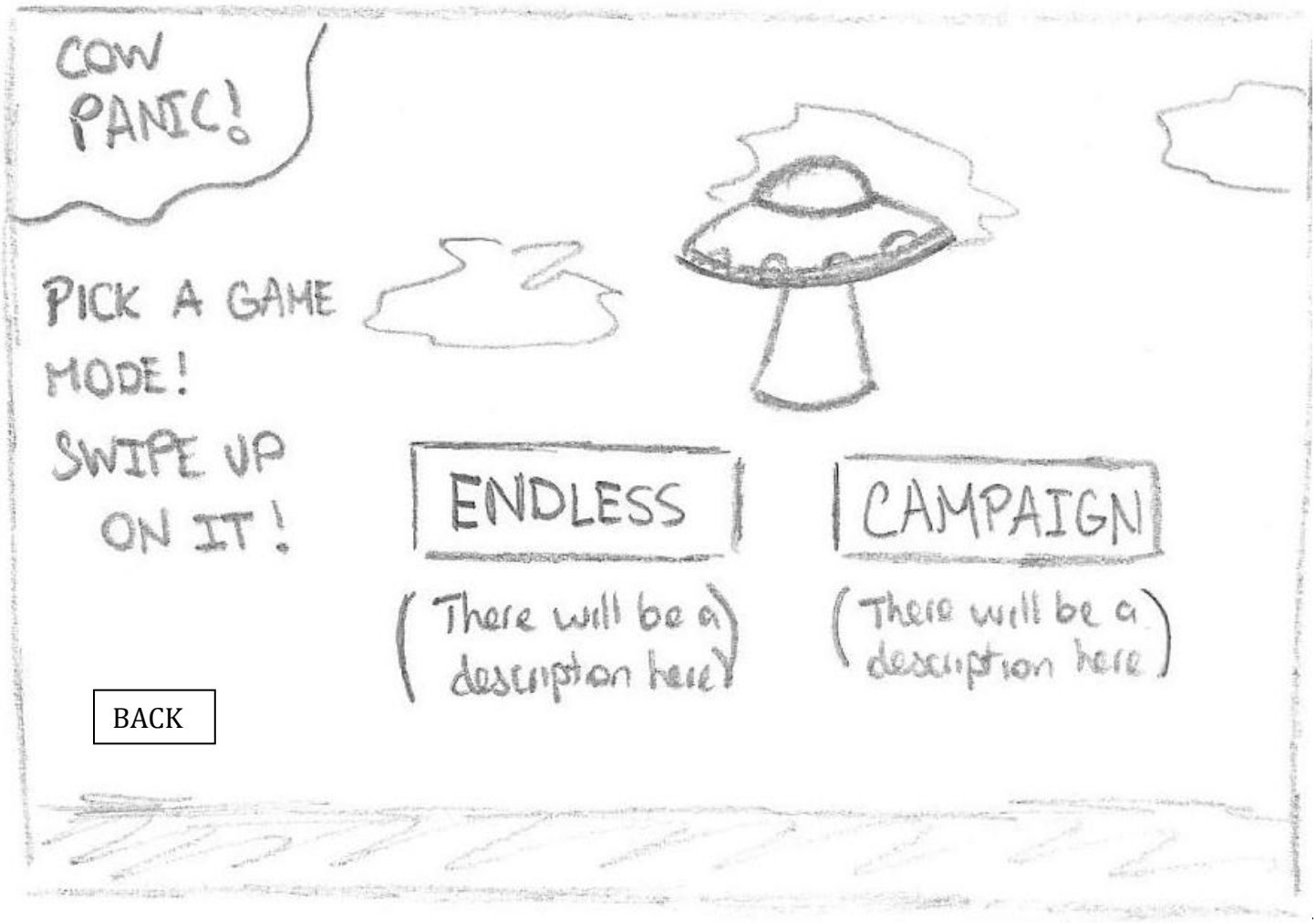
Type: Screen

Purpose: This screen is needed to meet requirement: menuPlay.

Description: Figure 3 shows the layout for this screen. This screen contains the Play menu, which allows users to pick from two of the game's modes.

The screen contains the following elements: - Boxes which, when swiped upward on by the user, cause an option to be selected and take the user to its corresponding screen.

Layout:



^This screen is now obsolete

Figure 3 – Play Menu Screen

Name: Settings Menu

Type: Screen

Purpose: This screen is needed to meet requirement: menuOptions.

Description: Figure 4 shows the layout for this screen. This screen contains the Settings menu, which allows users to adjust some of the game's basic options, like sound and music volumes.

The screen contains the following elements:
- Sliders that adjust the volume of sound effects and music.
- A toggle button that turns cow puns on and off. (Moo/Unmoo, respectively).
- A button that takes you back to the previous screen.

Layout:

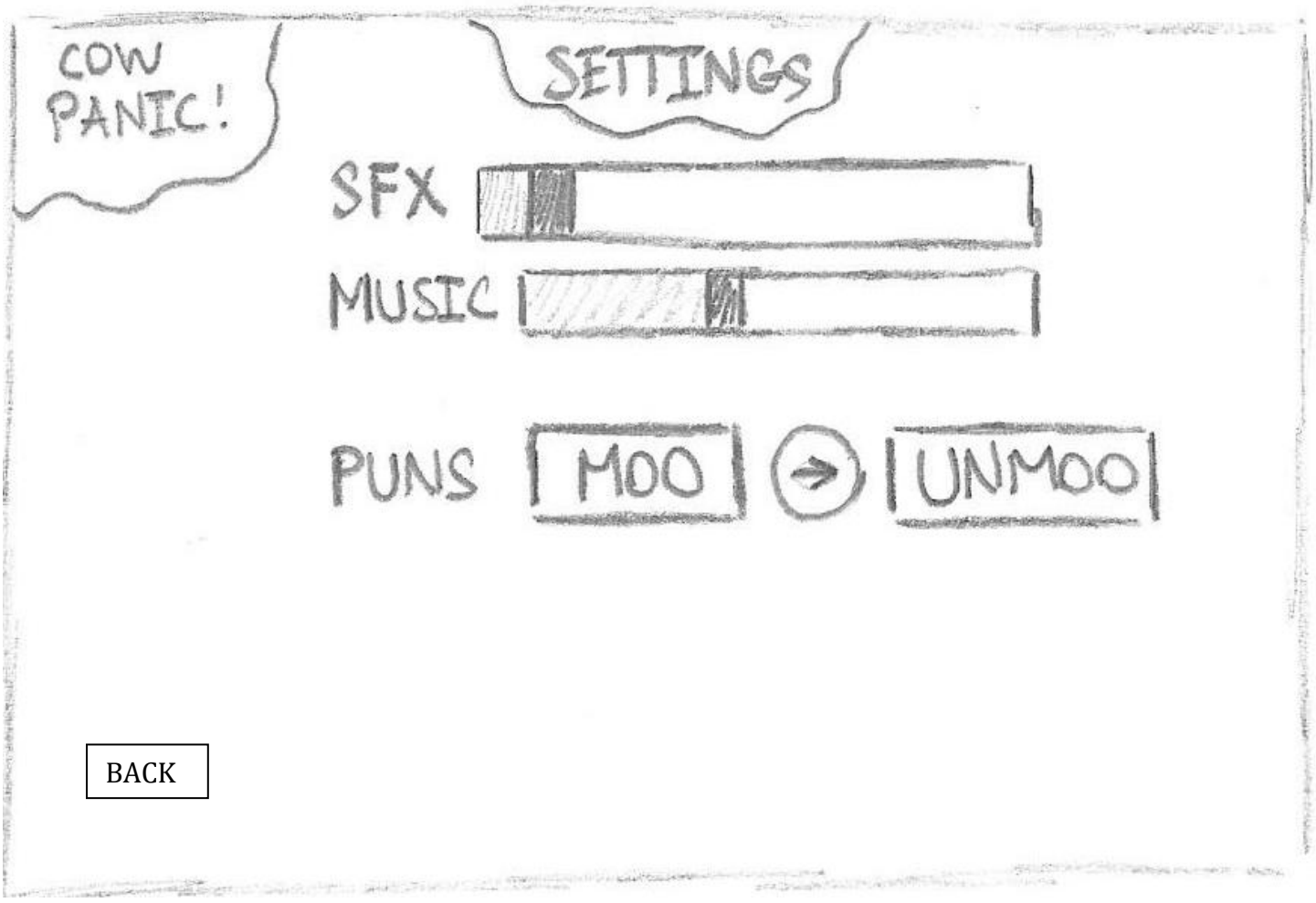


Figure 4 – Settings Menu Screen

Figure 7-4 – Template for Detailed Design for a Code Function

Name: checkCollision

Type: Function

Purpose: This function is used to check collisions between two objects

Parameters: The following parameters are used to call this function:

Name	*Data Type*	*Notes*
mask	*LayerMask*	*A LayerMask specifying which*

Return Type: returns a RaycastHit2D referring to the GameObject with which the implicit parameter has collided

Processing: Casts rays at each pixel position along the top of the object to the bottom and checks if those rays hit any object on a different layer

Name: Moove

Type: Script

Purpose: This script contains all functions for cow movement and abductions

Parameters: This script class has no parameters.

Return Type: This script returns no values.

Processing: It has a Boolean variable canMove that is checked every frame to determine whether the cow can move and checks collisions with the tractor beam to determine if it should stop moving.

Name: abduct

Type: Function

Purpose: This function is called by the UFO object when the player triggers an abduction and creates a tractor beam object with all necessary properties

Parameters: No parameters are needed

Return Type: returns no value

Processing: Create a GameObject called tractorBeam and add the necessary components, like BoxCollider and the tractor beam scripts BeamBehavior and BeamAnimator.

CI102

NOTE: There are not 2 data element revisions necessary because we only planned to have three data elements (two files) from the start, and both are still necessary for our plan for the game. Instead, we have removed more screen entities and added some descriptions of functions and scripts.

Figure 7-5 – Team Capability Assessment

	Jake Carfagno	**Drew Graham**	**Daniel Sipe**	**Josh Karmel**	**Jonah**	**Hugo Armella**
Programming – C#	3	3	2	2	2	2
Art – Pixel Art	3	2	2	3	2	3
Art – Moosic	1	1	1	3	2	1
Unity Experience	3	2	1	1	1	1

** The table values represent an assessment of team member capabilities. The values are:

1 – No knowledge or not much relative to the needs of this project
2 – Enough knowledge to accomplish part but not all of this project
3 – Knowledge probably sufficient for this project