DragonFruit UI Testing

Evaluation Methods Used:

- Think Aloud protocol
- Usability Inspection
- Heuristic Evaluation
- Design Walk-Through
- GOMS Evaluation
- Cognitive Walkthrough

Think Aloud Protocol

- Our test participants were given a task to complete, e.g. 'Build an apartment' and we asked them to think out loud about anything they were looking at, thinking, doing, and feeling while performing said task.
- This was very useful in determining confusion about the UI

Usability Inspection

- We had an expert (Dr Stuart Marshall and Roma) come in and evaluate our UI based on their knowledge.
- Feedback received from these sessions was used to improve/change the UI accordingly.

Heuristic Evaluation

- We had one person from the team go through each use case and check the design against a set of heuristics and rules.
- Feedback received from these sessions was used to improve/change the UI accordingly.
- The rules and heuristics used were:
    - 80/20 Heuristic
    - Fitt's Law
    - Hick's Law
    - Confirmation Heuristic
    - Flexibility-Usability Tradeoff
    - Recognition Over Recall
    - Form Follows Function
    - Progressive Disclosure

80/20 Heuristic

- 80% of users only use 20% of UI features
  or
- The critical 20% of features are used 80% of the time
- This heuristic was used to assess the value of UI elements
- The less-important 80% were ordered towards the bottom and the top 20% of buildings were placed at the top
- The importance was decided by our team and not users/experts so the data could be a little unreliable and biased

Fitt's Law

- The time required to move to a target is a function of the target size and distance to the target
- The smaller and more distant a target, the longer it will take to move to a resting position over the target
- The faster the required movement and the smaller the target, the greater the error rate due to a speed-accuracy tradeoff
- This law was considered when designing interactions that involve pointing
- Controls that are not frequently used were made more distant and smaller
- Controls are near and large when rapid movements were required and accuracy was important (e.g. build menu)
- Size of target (e.g. the 'X' button on build menu) is considered in the direction of movement
- Note: The law is logarithmic, a small increase in the size of a small object gives a large increase in usability
- Usability is increased by placing the 'X' button in a common place (reducing distance) and increasing its size

Hick's Law

- The time it takes to make a decision increases as the number of alternative increases
- We used this law to estimate how long it will take for a person to make a decision when presented with multiple choices. E.g. choosing a building
- UI interactions consist of four basic cognitive steps
  - Identify a goal
  - Assess the available options
  - Decide on an option
  - Execute the option
- Hick's Law applies to the third ("Decide on an option")
- If designing for time-critical tasks, the number of options must be reduced
  - Our menu has too many options (this is a bad thing seeing as there is a timer for the game)

Confirmation Heuristic

- This technique is used to prevent unintended actions by requiring verification of the actions before they are performed
- When a user tries to build a building, they are prompted with a dialog that displays information about the building and the consequences it has
- The user is asked to tap on the image of the building to confirm the action

Flexibility-Usability Tradeoff

- As the flexibility of a product increases, its usability decreases
- Flexible designs can perform more functions than a specialised design, but they perform the functions less efficiently
- We reduced the amount of noise (unnecessary information) on our UI and the amount of functions is minimal

Recognition Over Recall

- Memory for recognising things is better than memory for recalling things
- The UI has a menu and icons that conform to the people's mental models
- People are better at recognising things they have previously experiences than they are at recalling those things from memory
- For this reason, the UI provides memory cues such as menus, icons etc.

Form Follows Function

- Beauty in design results from purity of function
- Aesthetic considerations in design should be secondary to functional considerations
- The game would look nicer (better form) if we had a side view, but functionally, we needed a game that could be played from all sides.
- For this reason, we chose to have a top down game that could be seen from all angles

Progressive Disclosure

- A strategy for managing information complexity in which only necessary information is displayed at any given time
- This involves separating information into multiple layers and only presenting layers that are necessary or relevant
- We chose to hide information about buildings on the first build menu layer as it would become cluttered
- After choosing a building, more information is presented about that particular building
- This separation of information helps manage complexity without making the users confused, frustrated, or disoriented

---

Design Walk-Through Evaluation

- This whole method was based on scenarios and use cases
- We used an expert and our team were end-user representatives
- As a team, our goal was to explore design on behalf of the users
- We went through several workflow patterns of key uses and discussed merits and potential problems of the design.

GOMS Evaluation

- **G**oals, **O**perations, **M**ethods, and **S**election rules (GOMS)
- Example:
    - **Goal:** Build an apartment
    - **Operations:** Move, Drag, Tap
    - **Methods:** Move finger on tile, Swipe towards self, tap on 'Apartment' to build
    - **Selection:** Build by tapping finger
- The process involved a decomposition of all operations until they were primitive and KLM (Keystroke-level Model) parameters were used to estimate total time.

Cognitive Walkthrough Evaluation

- This method was based on task analysis.
    - Learning the system
    - Exploring action sequences
    - Identifying the sequence of steps necessary to accomplish a task
    - Identifying potential problems
    - Counting problems
- We asked 4 question on each user action
    - Will user try to achieve the right effect? (Selection)
    - Will user notice the action is available? (Visibility)
    - Will user associate the correct action with the effect? (Labelling)
    - Will user see progress is being made toward solution? (Feedback)

Some questions to think about:

Which features have been tested/ will be tested eventually?

-The whole GUI tested

How many user scenarios/ use cases have been executed?

-Opening up a build menu
-Building an object/building
-More than one person building at the same time
-Multiple build menus open
-Dragging/scrolling through build menu
-Simultaneous dragging/scrolling

How many features are stable?

-All but Simultaneous inputs on the GUI

Which features need more work?

-Simultaneous inputs on the GUI need more work. Currently, when a player holds his hand down on the GUI, all other GUI inputs are blocked. (GUI inputs exclude swiping on the game)

Are sufficient input combinations exercised?

-Yes, they have been thoroughly tested

Does the app give out correct error messages if the user does not use it the way it was intended to be used?

-Yes it does. The user is notified when there is insufficient employment, resources etc. and told that they cannot build.

Does the UI conform to the specifications?

-Yes. It gives a collaborative experience and is multi-directional and allows multiple players.

Are the features traceable to the requirement spec? Have all of them been covered?

-Yes, all have been covered.

Are the tests good enough? Are they finding defects?

-The tests found a lot of defects early on and we were able to change the GUI based on the feedback we received from test participants.