U08928 coursework submission part 2
Lee Hudson
09092543

Introduction:
After designing the reversi app in part 1 of the coursework I have successfully implemented the follow:

1. Basic functionality of the game
2. Have user selectable chip and board options
3. Have the program automatically flip the chips after a move
4. Have both single player and two player modes
5. Allow the player to pass
6. Have an optional timed mode with sound
7. Players selected from the contacts content provider with photos
8. Board created programmatically using a custom view and canvas
9. Store the high scores in a content provider and display them in a separate screen with photos.

As with some initial designs the implementation had some modifications. The finished app had the design on the following page. The differences between the initial design and the final design are as follows:

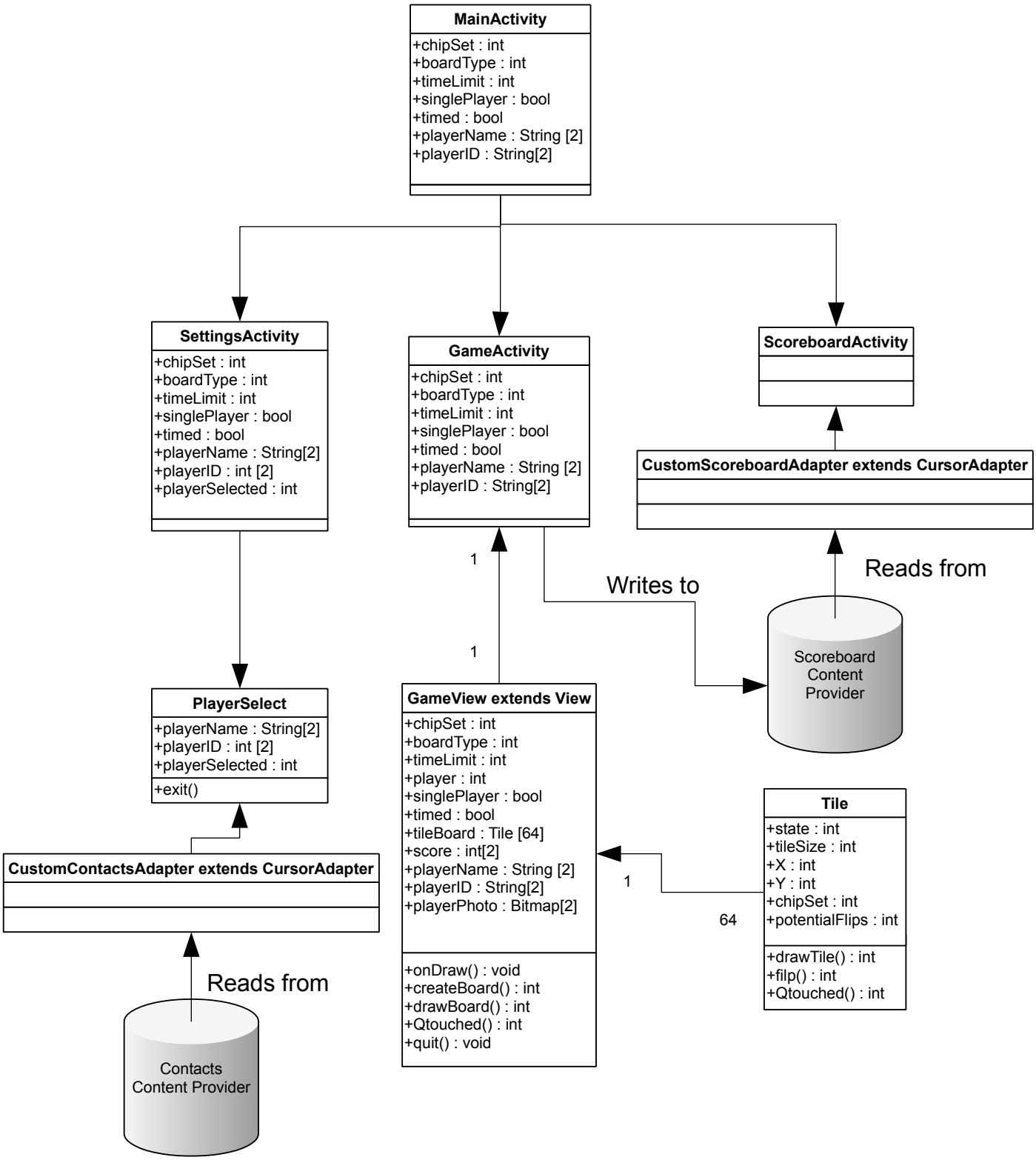
1. The player photo isn't passed between activities as a bitmap but is instead retrieved from the contacts content provider using the _ID field to retrieve the photo uri. This is a more efficient way of doing it.
2. Default settings are loaded in the main activity and passed back and forth between activities. So, the main activity has a value for a variable, this variable is now passed to say the settings activity. The settings activity can now modify this variable and pass it back. Likewise, the variable can be passed to the main game and when the game is finished or exited these values are returned to the main activity. This adds an element of intuition and continuity.

Description of the system:
The original design of the system has changed slightly. I was aware I was going to need to implement a custom content provider for the scores but I wasn't exactly sure how, here are the fields the provider has:

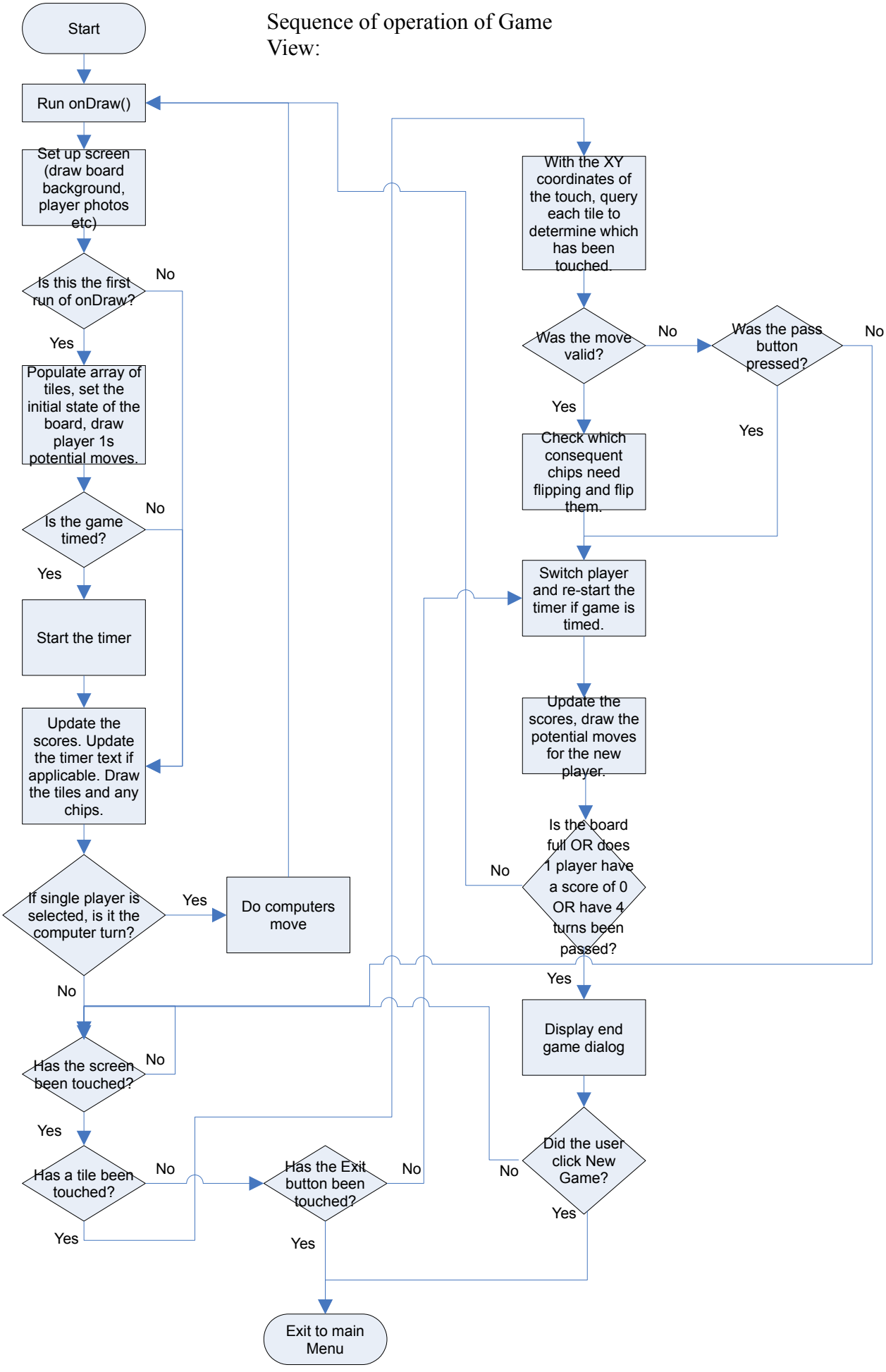
| | | |
|---|---|---|
| _id | - | This is just a unique identifier for the row |
| Name | - | This is the name of the player |
| Score | - | This is the score |
| Image | - | This is the uri in text form |

In order to display this in a list view on a separate screen I needed to create a custom cursor adapter as the base version of SimpleCursorAdapter doesn't support image views. The custom adapter created an image uri from the Image field (this field is just a string so it needed converting). The image uri was then used to create a bitmap which was passed to the relevant image view. This adapter was adapted further to display the contacts in the playerSelect activity in the same way.

Final Design:



Sequence of operation of Game View:



Testing:

The following test cases have been devised for the given activities:

Settings activity:

| **Description** | **Desired Result** | **Actual Result** |
|---|---|---|
| Put no value in time limit field | Time limit in the game reverts to default value | As desired |
| Check chipset 1 | Standard black and white chips selected | As desired |
| Check chipset 2 | Red and blue chips selected | As desired |
| Check board type 1 | Standard felt board selected | As desired |
| Check board type 2 | Wooden board selected | As desired |
| Check timed checkbox | Time limit appears in the game, game is played in timed mode | As desired |
| Uncheck timed checkbox | Game is not played in timed mode | As desired |

Game Activity:

| **Description** | **Desired Result** | **Actual Result** |
|---|---|---|
| Start game without accessing the settings activity | The game runs with default settings | As desired |
| Try and move on other players chip | Nothing happens | As desired |
| Try to do an invalid move | Nothing happens | As desired |
| Finish a game | Winning player is reported | As desired |

Score provider:

| Description | Desired Result | Actual Result |
|---|---|---|
| Add a player to the provider | Player is added to the provider and displayed in the scoreboard activity | As desired |
| Add a player to the provider that does not qualify | Player is not added to the provider | As desired |
| Add a player to the provider that is already in the provider | Player is added to the provider as a separate instance, previous instance is still present if applicable. | As desired |
| Add a player to the provider when the player is full | Player is added to the provider and the player with the lowest score is removed | As desired |

Note: The above tests on the score provider are very hard to do in a real game so the relevant method was forced into these conditions. The provider was also tested in normal operation too.

General interface:

| Description | Desired Result | Actual Result |
|---|---|---|
| Device is put into landscape mode | App is locked in portrait so should not respond to this aspect change. | As desired |
| Back button is pressed on any page | If page is not the main activity the app will revert back to its parent page. | As desired |
| Home button is pressed during a game | The game is paused until the user re-activates it or android os kills it | As desired |

Learning reflections and possible improvements:

In hindsight and with more time permitted there are a few areas I would have liked to improve on. First of all is the overall look and feel of the application. The main game view was done on a canvas which I liked, it's more flexible than the standard UI tool kit and if I used this for the rest of the app it may have made the app appear more consistent. I would also have liked to implement some animation to increase the responsiveness of the app, when the computer has a turn it happens so quickly its abit hard to follow.

I didn't do any of the calculation algorithms in separate threads as they were trivial enough not to have a big performance hit but if I had more time I would have liked to have done this. I could have implemented the canvas in a surface view but when the screen is only being re-drawn at a very low frequency (every turn or second if in timed) it did seem abit over kill. If the content was abit more dynamic, say a platform game I would have reconsidered.

Another improvement would have been to deal with screen rotation as opposed to just fixing it in portrait, there are a large amount of apps that are fixed in one orientation so I didn't feel too bad about doing this but if I were to publish the app I would want it to stand out. Currently the app uses a single XML layout file for each activity, these do work for the vast majority of screen sizes but were really designed for high resolution phones. If I had more time I would have created an XML for each screen size classification to make the application feel like it was designed with the users device in mind. I would have also liked to implement fragments for the settings page. On smaller screens showing the name and photo of the contact may appear abit cluttered, using fragments would mean I could put the photos on a separate screen for the smaller devices and had them on one screen for larger devices such as tablets.

I feel now that I have a good appreciation of how to program an android device. The concept of isolation between activities using intents was new to me but I have managed to implement this throughout the application and I now have a good understanding of it. The thought of using a custom cursor adapter to allow image views to be populated in a list view initially escaped me but eventually I figured it out, I didn't realise the standard cursor adapter didn't cater for this. The hardest part of the coursework for me was the content provider, after modifying the table definition I didn't know how to re-write the new definitions to the SQL database, I now realise I just had to increment the version number for the providers onCreate() to be called. Using a content resolver to access the provider was a tricky concept for me to grasp too but now I've done it a few times I'm happy that I understand it.

In Summary, there are many things to consider when programming for a mobile device, as it happens the app that I created isn't very demanding so resources don't appear to have been an issue although I still designed my app with those considerations in mind. The exercise over all has been very useful and I have really enjoyed making creating the application.