Interactive Farm Map Project
Chen Zheng Study Report

Chen Zheng
Student ID: 12087306

September 11th, 2012

1 Special Topic Tasks

The objective of the special topic is to develop an initial prototype that demonstrates the essential interactivity and functionality of the wall map. This includes:
• My overall propose of this project is to implement a MVC design pattern based website. For the View layout, I create some web pages as the interface to handle the web requests. For the Control part, my aim at implementing some nice and clear *javascript* with *google.maps.API* as a business manager to response those web requests. For the Model part, what am I propose to do is have a little try with *google.maps.FusionTablesLayout*.

1.1 Locate and load the correct farm map from GoogleMaps

• Load the *google.maps.com/api/js*

1.2 Draw a set of paddocks on the map, whose coordinates are stored in xml files (1 per paddock)

• Put an array of lat& lng into *google.maps.polygon′spath*.

1.3 Provide for each paddock to be labelled with up to three pieces of data (e.g. name, area, stock count) one of which may be an image

• I tried to use *groundLayout* to implement this functionality, but I failed. I do know I should call some *layouts* to do this, but I am quite confused now.

1.4 Paddock data is persistent and updateable (stored in a database)

• I think we can use *FussionTableLayers* which is still experimental right now, but this object allows us to construct this object such as: *google.maps.FusionTablesLayer* and put our *querysentance* into its *optionsproperty* like this:

```
var layer = new google.maps.FusionTablesLayer({
  query: {
    select : 'attribute.Name',
    from : 'table.Name',
    where : 'trackName=Track_P1.xml'
  },
});
```

So if we can, we are be able to query and update data with *FusionTablesLayer*.
Another option for this is to create an our own *database* to provide data persistance.

1.5 The farm map may be updated by selecting a function on the LHS of the screen (running a database query), e.g. showing current location of cows

• Similarly, I prefer to use FusionTableLayer to do the data persistance function. And encoding the query things into the script file.

a. Some paddocks may be highlighted (e.g. border colour changed or area filled-in).
   • For these functions, we can call a method to modify the *property.fillColor* or other *properties* of the polygon DOM element.

b. Paddock labels may be changed to show different data.
   We can create some buttons which are associated with some methods to those query functionality.

1.6 If a paddock is touched (or clicked) on screen

a. The paddock is highlighted.
• The *google.maps.polygon* object has some events which can be listened. So we can add listener to listen *mouse events* to make this polygon clickable.

b. Paddock data is displayed in a box in the LH corner of the screen.
• We can create a $infoWindow$ to display these data which comes from the $querying\ of\ database$ *or* $FusionTablesLayer$.

c. This data is updateable.
• If we use these databases, we can update them in database. Then just redo the querying things and the $Website$ will show these data.

2 The initial prototype:

2.1 Should be implemented for paddocks, but provision made for its extension to other farm entities e.g. buildings, or tracks

•

2.2 Should be implemented to show (and update) location and grazing history of stock but should be easily extendable to other queries

•

2.3 Should support touch screen interactions

•