GPS Drawing 3:
Android Continuous Location
Mobile and Ubiquitous Games
ICS 163
Donald J. Patterson
GPS Drawing - Changes to making the single location continuous

- **MANIFEST**
  - **PERMISSION**
  - **GOOGLE PLAY VERSIONING**

- **BUILD UI IN XML**
  - **BUTTON** → **LAUNCH CONTINUOUS UPDATES**

- **LOCATION MANAGER** / **LOCATION REQUEST**
  - **ONCREATE**
    - **SUPER.ONCREATE( )**

- **INTERFACES**

- **CONNECT LOCATION → GOOGLE PLAY**

  - **CALLBACK**
    - **SUCCESS** → **ASK FOR LOCATION**  SET A FLAG SAYING WE ARE CONNECTED
    - **FAIL** → **INVOKE PLAY RESOLUTION**

- **LIBRARY FOR UPLOAD**

  **UPLOAD** (GROUP NAME, DRAWING NAME, STROKE NAME, R,G,B, LIST<POINTS>, POINTS)
• Instead of getting location on the Play connection call back, you can set a variable. Then a button can be used to initiate the continuous location feed.

```java
@Override
public void onConnected(Bundle connectionHint) {
    // Display the connection status
    Toast.makeText(this, "Connected", Toast.LENGTH_SHORT).show();
    mLocationClientConnected = true;
}
```
GPS Drawing - Changes to making the single location continuous

- Dynamically assign a function to a button
- When clicked it launches the location call backs

```java
Button startButton = (Button) rootView.findViewById(R.id.button1);
startButton.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        if((parent.mLocationClient != null) && (parent.mLocationClientConnected) && (parent.mLocationRequest != null)) {
            parent.mLocationClient.requestLocationUpdates(parent.mLocationRequest, parent);
        }
    }
});
```
GPS Drawing - Changes to making the single location continuous

- The location callbacks are defined by the LocationListener interface

```java
public class DevonIsAwesome extends Activity implements
    GooglePlayServicesClient.ConnectionCallbacks,
    GooglePlayServicesClient.OnConnectionFailedListener,
    LocationListener{
```
GPS Drawing - Changes to making the single location continuous

• Which defines the onLocationChanged callback.
• In that function we update our U/I

```java
@override
public void onLocationChanged(Location location) {
    String display = "("+location.getLatitude()+","+location.getLongitude()+")";
    this.lastLocation = display;
    updateUI();
}
```
GPS Drawing - The flow for the assignment

```
LAUNCH
CREATE LOCATION OBJECTS

DEFAULT GROUP NAME - DRAWING NAME

START CONTINUOUS UPDATES

INTERNAL VARIABLE
CURRENT COLOR (0-255, 0-255, 0-255)

ON TOUCH DOWN
CREATE AN INTERNAL STROKE NAME 0 255 0

• UPDATING UI

• STORING LOCATION IN AN LIST

• UPDATE BUFFER SRF DISPLAY

ON UPLOAD BUTTON
PEN UP

UPLOAD DATA
CLEAR BUFFER
```
GPS Drawing - Changes to making the single location continuous

- In the final assignment you will have to keep track of three additional things (primarily):
  - A color that the user selects
  - Whether the pen is up or down
  - A collection that holds the points that you are observing when the pen is down

- The final bit about how to upload is on its way