GLASS

Christopher Bartholomew
cbartholomew@g.harvard.edu
CS50 Seminar: Breaking Through The (Google) Glass Ceiling

The ultimate success of any wearable technology depends on the applications that are available.

As glass developers the challenge is to create apps that are more *convenient* and *innovative* than our own mobile device apps.

If you completely wiped your mobile device, what would be the first five applications that you would download? Why?

Agenda

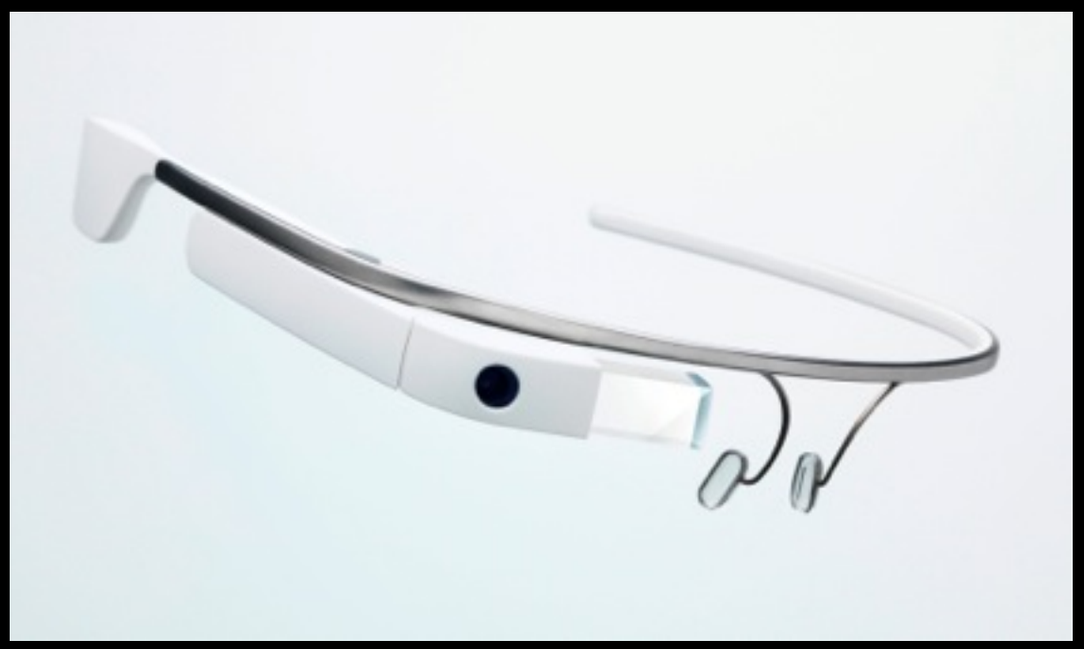
- what is glass?
- the glass timeline
- design patterns for glassware
- mirror api vs glass developer kit (gdk)
- pro-tips
- resources

WHAT IS GLASS?

the user interface http://youtu.be/4EvNxWhskf8

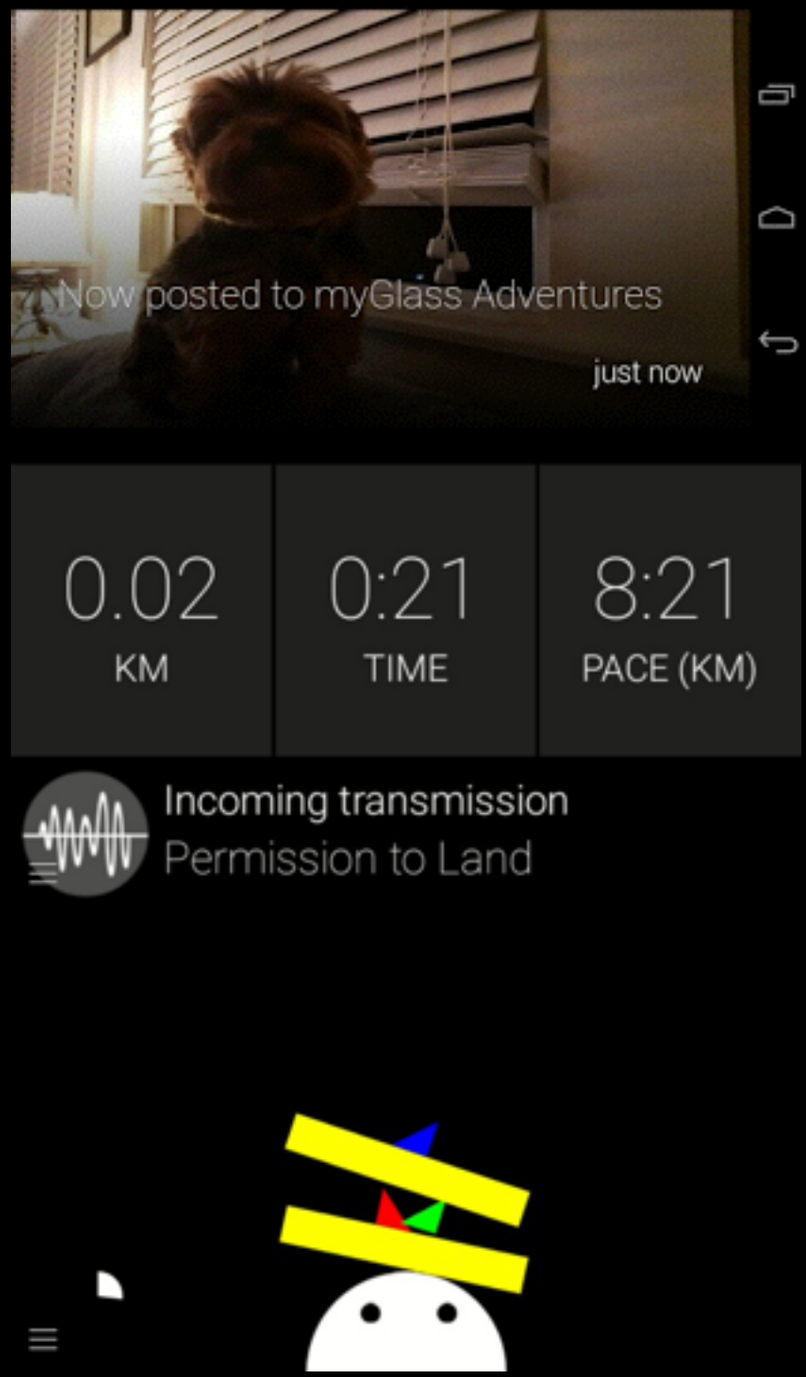
WHAT IS GLASS?

- display
  - 640x360
- camera
  - 5 Mega Pixel Camera / 720p Video Recording
- wifi / bluetooth
  - 802.11 b/g
- storage
  - 12 GB (not including OS)
  - 2 GB of RAM
- goodies
  - 3 axis gyroscope
  - 3 axis accelerometer
  - ambient light sensing and proximity sensor (wink)
  - bone conduction audio transducer

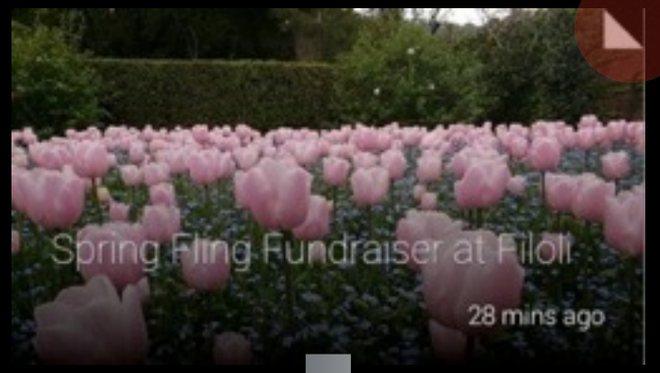


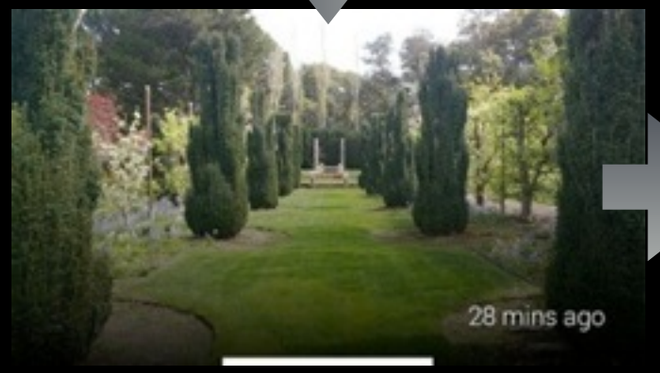
TIMELINE CARDS

- static card - displays text, html, images, and video
  - *can be used to invoke a live card or immersions.*
- live card - displays cards that that are important at the current moment, rendered.
- immersion - displays android activities that take over the timeline experience.



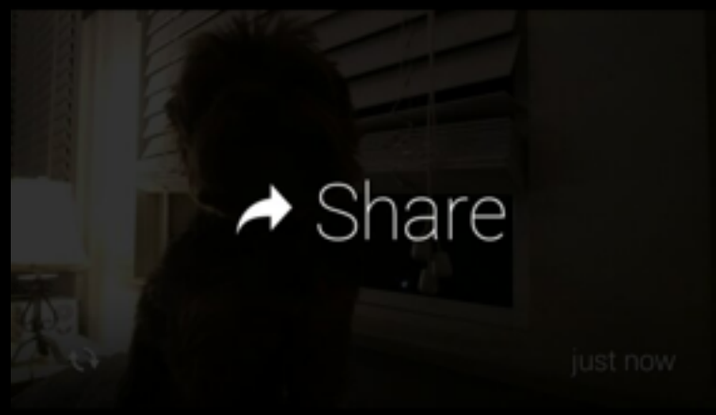
TIMELINE CARDS





- static cards can also contain a “bundle” of cards, for example, a photo album.

- static cards can also have menu items that invoke some action.



DESIGN PATTERNS

- Periodic Notifications

  - uses mirror api web services or android background services to push notifications on the timeline.

    - mirror api can use any language to create a web service interaction.

    - android background services uses Java per the Glass Developer Kit (GDK)

- example: new york times

DESIGN PATTERNS

- Ongoing task
    - This is a card that is constantly running in the background. Users can leave the card and explore the timeline while the task continues to run in the background.
        - can only be implemented using the Glass Developer Kit (GDK).
        - can take full advantage of glass hardware.
        - android GDK is Java only.
- example: stop watch / zombies, run!

DESIGN PATTERNS

- Immersion
    - immersion will consume the entire timeline experience (also known as an android activity)
        - can only be implemented using the Glass Developer Kit (GDK).
        - can take full advantage of glass hardware.
        - android GDK is Java only.
- example: games!

INVOCATION METHODS

- A fancy term for “how do I start my application?”
  - Different design patterns will leverage different invocation models. For example *Periodic Notifications*, do not use any invocation methods.
- “ok glass” voice menu
- contextual voice or touch menu on a timeline card

MIRROR API

- before the GDK, mirror was used as the primary application interaction.
- uses primarily backend web services to accomplish simple tasks such as timeline subscriptions, content sharing, and voice text.
- glassware using mirror api can be written in almost any language.
- glassware that *only* uses mirror api will not have the ability to use voice invocation methods nor the hardware goodies such as the gyroscope and accelerometer.
- can create glassware without a physical device in hand.
- requires knowledge of oAuth2

GDK

- glass developer kit is in essence an extension to android development.
- can only be used with java.
- developers have full access to *all* native hardware features - including gyroscope and accelerometer.
- primarily used for voice invocation abilities, live cards, and immersive applications.
- requires that the users has a device in hand.

GDK vs MIRROR

- GDK
  - Pros: straight forward for android developers (beginner or professional), ability to use all native hardware, opens the door for interactive glassware, offline functionality, and lots of documentation.
  - Cons: only available in one language: java, Integrated Development Environment (IDE) setup time can take awhile, many considerations such as battery life and memory usage.

GDK vs MIRROR

- Mirror API
  - Pros: can use almost any language, requires minimal development experience, uses straight forward web architecture to integrate with the cloud, doesn't require having a device, and many tools to help the user create content including lots of documentation.
  - Cons: oAuth2 knowledge (i.e. sign in with Google Account) no hardware interaction (mirror only), all web-based, requires knowledge of basic HTTP concepts such as POST/GET/PUT/DELETE, and **no** offline functionality.

CAN'T DECIDE?

- DON'T WORRY! YOU CAN USE BOTH!
- for example, let's say you create game using immersion via GDK. You also create a leader board that's in the cloud. Using the mirror api you can send your score to your "leader board" web service so it can be displayed.
- ***use what's best for your application*!**

PRO-TIPS

- Utilize the glassware flow designer to create your application timeline
  - https://glassware-flow-designer.appspot.com/
- Utilize the glassware playground to review and create content that is displaying on your application
  - https://developers.google.com/glass/tools-downloads/playground
- PDF Handout

RESOURCES

- Google Developers: Glass
  - https://developers.google.com/glass/design/principles
  - https://developers.google.com/glass/design/patterns
  - https://developers.google.com/glass/design/style
  - https://developers.google.com/glass/develop/mirror/authorization
  - https://developers.google.com/glass/develop/mirror/index

RESOURCES

- GDK Quick Start Guide
  - https://developers.google.com/glass/develop/gdk/quick-start
- Android Tutorials:
  - http://youtu.be/Z149x12sXsw
- Google Style Guide: Java
  - https://google-styleguide.googlecode.com/svn/trunk/javaguide.html