Crosswalk on IoT

Kenneth Christiansen, Sakari Poussa, Tiago Vignatti
Session goals:

- What IoT means to Crosswalk from the **graphics perspective**
- Introduce **a new graphics architecture for IoT**
- Next **challenges**
Tizen graphics, IoT and Crosswalk
Tizen Graphics

- **Tizen** is not much different than traditional Linux distros:
- **In short**: kernel Linux + GL driver + X11 or Wayland
  - Native App: toolkit (EFL or Qt)
  - Web App: runtime (WebKitEFL or Crosswalk)
- **GL graphics context requires** **several megabytes!** (sorry, no reference)
  - Problems on constrained platforms:
    - **memory allocation**: GPU driver resources, texture storage, double-buffering etc
    - **memory bandwidth**: texture upload of bitmaps
IoT

• IoT display-based devices:
  – medical monitors, smartwatch, wrist, etc

• hardware are not very capable:
  – CPU < 1 GHz, memory < 512 MB, no GPU

• system is somewhat simple:
  – e.g. one fullscreen web app at each time
    • simple window management
    • simple UI
Crosswalk (1/2)

- Crosswalk is based on Blink and Chromium
- It implements Tizen Web APIs for system control
- Chromium has a new platform backend system called Ozone:
  - Crosswalk on Tizen IVI uses Ozone-Wayland
  - Ozone-Wayland implements Wayland platform for Chromium
  - There are other Ozone implementations like KMS/DRM, caca, testing, etc
Crosswalk (2/2)

- We believe that Crosswalk could encompass all IoT needs!
  - *Web is the whole system*
  - a lean graphics architecture is required though
a new graphics architecture for IoT
Solving Tizen Graphics issues for IoT

- Graphics architecture for IoT has the desired features:
  1. Able to run in constrained platforms
  2. Simple window management
  3. Simple UI

- Solution:
  - remove the window system and toolkits
    - why we'd need it given that apps are fullscreen and Web based?
  - choose renderer method
    - e.g. using software rendering instead GL we potently could reduce memory problems
How Chromium helps? (1/2)

• Ozone:
  - Chromium Ozone backend system lets us to easily **switch the platform implementation**
  - We'd use Ozone KMS/DRM through software composing backend for constrained platforms
    • Ozone KMS/DRM uses double-buffer Skia surfaces, so it's quite capable
How Chromium helps? (2/2)

• Aura:
  – Aura is the UI framework for basic window and input events
  – Aura windows only have one graphics surface layer each (so window management is not really needed at the window system level!)

• Views:
  – Views is Chromium's internal widgets toolkit based on Aura
  – If desired, more complex windows decorations can be done using Views (no external graphics toolkits are needed!)
Crosswalk graphics architecture for IoT
Conclusion

• The new architecture is meant for IoT
  - constrained hw platforms where the Web takes over the whole system

• Less overall complexity due code reduction
  - Easily we save at least 1 million LoC (window system + toolkits)

• Proof-of-concept:
  - https://github.com/tiagovignatti/crosswalk/commits/embedded
  - Using Tizen Common (“Generic”)
Next Challenges
Next Challenges

• Drawback: no Native App option anymore for Tizen
  – Everything goes through Chromium architecture
  – What about NaCl?

• Are we fine with Web performance for the UI?
• Send code to upstream Tizen and Crosswalk
Questions? Thank you!