Panda3D: iOS Support

Motivation

Panda3D is a mature 3D rendering engine both in age and functionality, addressing the need for the rapid prototyping of games while still providing stability for large projects. As a result of this maturity, however, Panda has been slow to take advantage of newer platforms and devices, with no support for iOS and only experimental support for Android. Adding support for iOS will make the engine much more appealing to new developers, while also creating the opportunity to update more antiquated areas of the engine. Many of these features will aid in future work on Android support as well.

Project Goals

iOS support for Panda's new CMake build system
Although makepanda is Panda's primary build system, the CMake build system has been making great progress and is close to completion. Given makepanda's imminent deprecation, supporting it would take an disproportionate amount of time when compared to the amount of use it will see, so I will be focusing on CMake support instead.

Adding iOS support to the current CMake system would roughly involve:

- Updating the CMake script to support CMake 3.14's integrated iOS support.
- Adding the ability to build for any combination of supported architectures.
- Implementing support for cross-compiling Python as well as essential third party libraries using rdb's panda3d-thirdparty repository. I have managed to get this partially working already.

Support for dedicated Panda apps (with the potential to embed Panda inside existing apps as a stretch goal)
Panda is used to having control over the application life-cycle and event loop, so if the goal of making Panda embeddable into existing apps is to be achieved, the Panda startup process will need to be customized slightly for iOS applications. Early on in the development of this project, a Panda app will simply do the job of creating a `UIApplication` itself, since this is a relatively easy first feature to implement. But by

the end, I hope to support both Panda-exclusive apps as well as embedding Panda inside existing apps. This is not a requirement for the completion of the project, but it is definitely something I would like to finish.

To achieve this end-goal, a `PandaGLViewController` will be created to manage the Panda lifecycle in an iOS-friendly way. Rather than applications starting directly through Python, a "wrapper app" will be used to run Python games. This would simply be an app containing this view controller filling the entire screen. A `PandaGLViewController` will be initialized by pointing it to a script contained in the app bundle. If an instance of this script is not yet running, it will invoke it in a separate thread, similar to the Android port. Panda will initialize normally, and the `GraphicsWindow` instance will send a pointer to the `UIView` it created over to the main thread so it can be attached to the view controller.

A `PandaGLViewController` is meant to be analogous to a `GraphicsWindow`. Creating the first instance of a `PandaGLViewController` will automatically initialize Panda, and any additional instances will act like additional windows being opened. This opens the door to advanced applications that integrate `UIKit` controls and the use of multiple view controllers. This will likely not be of use to games or other dedicated Panda apps, but it means existing apps will easily be able to integrate with Panda.

Basic touch support (multi-touch as a stretch goal)
At the moment, Panda only supports single-pointer inputs. At minimum, I aim to have touch input on iOS be on par with other platforms, with touches being translated to mouse clicks as appropriate.

With the prevalence of multi-touch gestures like pinching to zoom or using two fingers on the screen at once for on-screen controls, multi-touch support will be a vital part of Panda's success on iOS. rdb has already begun writing a spec for multi-touch support, and has begun implementing it on the `multitouch` branch. If time permits, I plan on continuing that work and implementing the most important features than pertain to iOS, but this is by no means a requirement for the completion of the project.

Xcode project templates (stretch goal)
Xcode includes the ability for projects to define custom templates for apps, so it would be nice to be able to do so for Panda. It would include a Python and C++ template, making it easy to for newcomers to start up an iOS project. It would also

streamline deployment to the app store. A developer should also be able to change the default Python app location to their existing codebase. This functionality would be added only if the other base features are implemented in time.

Documentation and unit tests for all of the above
All of the above will need to be documented in Panda's manual. I plan on writing the documentation for any new features that are implemented and a small tutorial showing new users how to set up their projects for iOS.

In summary, the minimum work completed at the end of the coding period will be:

- CMake support
- Panda apps through managing a `UIApplication`
- Basic touch support
- Documentation and unit tests

Timeline

Below is my expected timeline for completion. Because this is such a large undertaking, it is quite difficult to know what issues I will run into at different stages of the project. As the project progresses, this schedule will be adjusted as appropriate. Additionally, this is meant as more of an outline for what order I will complete the project in rather than a true week-by-week breakdown (as in, some weeks may blend into each other). Regardless, I will have at least one deliverable at the end of each week to be evaluated.

May 6 - 27: Research, planning, and experimenting
During the community bonding period, I will attempt to re-familiarize myself with the ins and outs of Panda's codebase, and how each component interacts. I want to be sure I will be spending time implementing the features I would like to, instead of second-guessing myself later about the design choices I have made. I will create a list of potential pain points for iOS specifically while doing this, so I will have a better idea of what bugs I may need to fix. During this time I will also be setting up VMs for testing on various macOS hosts and iOS versions.

A few things other things to do during this period include:

- Experiment with Apple's threading systems (comparing POSIX threads with Apple's GCD system).

- Wrap my head around the features added in the recently merged `input-overhaul` branch to see how multi-touch can fit into it.
- Fiddle with CMake's ability to support cross platform builds, since CMake is unable to build for multiple platforms at once (this will be an issue when we need to run `interrogate` on the host).

May 27 - Jun 2nd: Add ability for panda3d-thirdparty to build third party libraries for iOS.
I will be ensuring that the most important third party libraries are compilable for iOS. Additionally, I would like to add the ability to build a custom Python as well, since there is no official iOS distribution. Two viable open-source projects to achieve this are:

- https://github.com/kivy/kivy-ios - Used by Kivy to run on iOS devices. It is able to build both Python 2 and 3, but may require some modification to support dynamic library loading.
- https://github.com/python-cmake-buildsystem/python-cmake-buildsystem - A project that allows Python to be built using CMake instead of its usual build tools. Contains a lot of extras that are not needed for iOS development, however, so this may not be the best choice.

Deliverable: Working compilation for the most important third party libraries on iOS, including Python.

Jun 3rd - Jun 9th: Update Panda's CMake script to support iOS builds
During this week, I will make sure that Panda's CMake script is able to take advantage of CMake 3.14's new iOS, tvOS, and watchOS support. This will involve ensuring that `interrogate` is successfully built on the host architecture so it can still be run on the Python files. The generated C++ files will then be compiled for the target ARM architecture. Additionally, the user should be able to set the target SDK version and architectures manually through configuration variables.

Deliverable: A working CMake script that allows for iOS compilation on a Mac system, given the requisite third party libraries.

Jun 10th - Jun 16th: iOS backend design
This week I will be consulting with mentors about the design of the backend, and tinkering with a few prototypes that will help me narrow down my goals. The supporting `display` classes (`GraphicsWindow` , `GraphicsPipe`, etc.) will be prefixed

with `EAGL` and be compatible with iOS, tvOS, and watchOS, all of which use the same underlying `UIKit` framework. While iOS support is the main priority, I also want to open the door for tvOS and watchOS support down the road.

At this point, the design will assume that Panda will have total control over the application life-cycle, but it should also be flexible enough to be changed in the coming weeks if no roadblocks arise.

Deliverable: A design and plan for implementing the iOS backend.

Jun 17th - Jun 23rd: Begin implementation
At this point, I will have a mostly complete design of how the iOS display backend will look. Next, I will begin the implementation, working closely with mentors to ensure that any potential design flaws or shortcomings are caught early. By the end of this week I would like to have a mostly finished base implementation of the iOS backend, with support for running `pview` as an iOS app at the very least. Additionally, some unit tests to check if a resize of Panda's `UIView` leads to a change in the `GraphicsWindow` size, for instance, would also be quite useful.

Deliverable: A baseline implementation of the backend that will at least show that I am going in the correct direction.

Jun 24th - Jun 30th: Python support, and the beginnings of `PandaGLViewController`
This week I plan to get Python scripts working as well. This will not require many changes to the display backend that I will have already implemented, but there will certainly be a few quirks that will need to be ironed out.

Additionally, I would like to begin working on the design of how an embedded Panda app would work, and determine what limitations may be met when trying to meld iOS's requirements with the existing design of the engine. The most likely scenario is that Panda's event loop and OpenGL context will live together in a separate thread. Because UIViews cannot be accessed from outside the main thread when attached to the view hierarchy, some extra considerations may need to be made.

If I begin having difficulties implementing the base level of iOS support, I will skip the addition of the embedded support in the interest of having a better quality backend that can be built upon in the future.

Deliverable: Working Python support, along with a design for `PandaGLViewController` if time permits.

Jul 1st - Jul 7th: iOS backend bug-stomping and polish, `PandaGLViewController` implementation
The iOS backend may be pretty much feature-complete at this point, but there will likely still be some oddities. I am taking this week to fix any loose ends that may have popped up, and to complete any features that may not have been implemented the previous week. This includes `PandaGLViewController` with embedded app support if the decision is made to proceed with it.

At this point, things should be mature enough to begin writing documentation as well.

Deliverable: A more polished backend, along with a working `PandaGLViewController` if time permits.

Jul 8th - Jul 14th: Basic touch support, continuing bug fixes for iOS backend
This week I would like to implement basic single-touch support, which will simply translate touch events into mouse events. I will also continue fixing issues, writing unit tests, and writing documentation.

Deliverable: Basic touch support.

Jul 15th - Jul 21st: Design for multi-touch support
At this point I would like to turn towards further fleshing out the design of rdb's `multitouch` branch. By the end of this week I would like to have a mostly-completed design of how multi-touch support will integrate with the rest of the engine (the event system, `MouseWatcher`, input-overhaul, etc.).

Deliverable: A design for multi-touch support based on the already existing design document.

Jul 22nd - Jul 28th: Base implementation of multi-touch
Since I will have a somewhat-final design at this point, I can use this as a roadmap to begin implementing multi-touch support.

Deliverable: A base multi-touch implementation that offers compatibility with existing mouse-based solutions.

Jul 29th - Aug 4th: Continued multi-touch work
This week I will continue work on multi-touch support. By the end of the week I hope for multi-touch to be usable from both C++ and Python.

Deliverable: Mostly-finished multi-touch support.

Aug 5th - Aug 11th: PStats, project templates, deployment (stretch goal)
This week I will be doing some streamlining of the development experience. This means easily supporting PStats and Xcode's built-in profiling and debugging tools, as well as creating project templates for both Python and C++ Panda projects. The installation of these project templates should be able to be turned off from the installer. Additionally, I would like to make sure apps can be properly archived by Xcode for release. Ideally, a minimal level of support for deploy-ng will be included along with this.

If the multi-touch work is beginning to run behind schedule, I will expand its work into this week and drop the above features.

Deliverable: Listen for a PStats client to connect over the wire to the development host, project templates, working deploy-ng and Xcode deployment.

Aug 12th - Aug 19th: Remaining Unit Tests and Documentation
At this point, all features should be implemented. In terms of documentation, I will be creating a guide for building the engine for iOS, an entry for basic usage in the manual, and detailed documentation for any other features not already documented.

Deliverable: Completed documentation and tests for all implemented features.

Previous Panda3D Contributions

I have previous experience contributing to Panda3D:

- Initial support for the new CMake build system on macOS
- Workaround for a macOS Mojave OpenGL regression

- Fix for an issue that causes a window to not close properly when destroying ShowBase
- Prevent a deadlock when using SIMPLE_THREADS on macOS
- Use CVDisplayLink on macOS to sync to the display

I have submitted several bug reports in the past as well.