

그래픽스 언어 대모험

키네티짱

발표 이유

- গত과목 그래픽스를 수강하세요
- 님들은 저처럼 삽질하면 안됩니다
- 강 암전히 C++을 써보세요
- C++을 안쓸거면 잘 들으시면 됩니다.

해야되는거

- OpenGL
- GLUT 라이브러리 - 티팟을 그려야하므로

하라고 한거

- C++ / OpenGL
- 유니티 오브젝트 하이어라키
- 물체 움직이고 드래그하기
- 점 잇기
- 투명 박스 만들기
- 레이 트레이싱 (광자를 추적하여 개 사실적인 이미지 뽑기)

고려는 해본거

- Python
- Java
- Ruby(WebGL)
- C#
- Clojure
- OCaml
- Racket
- Swift
- Objective-C
- Scala
- Rust
- Nim
- Common Lisp
- F#
- Julia

진짜로 코딩해본거

- Python
- Rust
- C#
- Nim
- Swift
- Common Lisp

이따거로 한 이유

- 조교를 괴롭히고 싶어서

첫번째 - C#

The screenshot shows the OpenTK website homepage. At the top is a navigation bar with links: Home, Project, Documentation, Forum, Issues, Blogs, Gallery, and Search. The main content area on the left describes OpenTK as an advanced, low-level C# library wrapping OpenGL, OpenCL, and OpenAL, suitable for games and scientific applications. It highlights three key features: Rapid Development (strong types and inline documentation), Easy Integration (standalone or integrated into Windows.Forms, WPF, or GTK#), and Total Freedom (MIT/X11 license). To the right of these features are two buttons: 'Download OpenTK' and 'Source Code'. Further right is a login section with a 'Log in' button and links for 'Create account' and 'Reset password'. Below the login section is a 'Recent activity' list showing recent forum posts. At the bottom, there are three columns: 'Feature Highlights' (rich type-safe bindings, flexible GUI options), 'Project Blog' (recent announcements like Vulkan support), and 'Quick Links' (source code, bug reports, frequently asked questions, etc.).

OpenTK *Home of the Open Toolkit library*

Home Project Documentation Forum Issues Blogs Gallery Search

The Open Toolkit is an advanced, low-level C# library that wraps OpenGL, OpenCL and OpenAL. It is suitable for games, scientific applications and any other project that requires 3d graphics, audio or compute functionality.

Rapid Development
Strong types and inline documentation improve your code flow and help you catch errors sooner.

Easy Integration
Use the toolkit standalone or integrate it into your Windows.Forms, WPF or GTK# application.

Total Freedom
The Open Toolkit is distributed under the permissive MIT/X11 license and is absolutely free.

[Download OpenTK](#)

[Source Code](#)

Username
.....
[Log in](#)

- [Create account](#)
- [Reset password](#)

Recent activity

- [Help with converting mouse coords to world coords](#)
- [Direct State Access](#)
- [OpenAL "Invalid value" error on Buffer<float>](#)
- [Matrix4 rotation / translation / scale](#)

[more](#)

The Open Toolkit runs on many operating systems and can be used by every Mono/.Net language: C#, VB.Net, C++/CLI, F#, Boo and many more.

Feature Highlights

- **Rich, type-safe bindings**
Support for the latest versions of OpenGL, OpenGL|ES, OpenAL and OpenCL with automatic extension loading, error checking and inline documentation.
- **Flexible GUI options**
Cross-platform GLControl (Windows.Forms), GLWidget (GTK#) and WPFControl classes. Native, high-performance GameWindow designed

Project Blog

- [Khronos group announces Vulkan](#)
- [The Open Toolkit library 1.1.4](#)
- [The Open Toolkit library 1.1.2](#)
- [The Open Toolkit library 1.1.1](#)

[Subscribe to OpenTK news](#)

[View older news items](#)

Quick Links

- [Source code](#)
- [Bug reports](#)
- [Frequently Asked Questions](#)
- [OpenGL reference](#)
- [Project gallery](#)
- [License](#)

Commit log

- [Merge pull request #202 from Frassle/issue201](#)
- [\[Input\] Legacy keyboard](#)

- OpenGL을 쓰려면 OpenTK라는 라이브러리가 필요
- C#으로 3D 뭔가 만들거면 강 DX를 씹시다

문제점

- 언어는 내가 본 언어중에 최고의 완성도를 지님
- GLUT 지원이 끊겨서 쓰기 힘들다
- 티포트를 못그림 TT
- GLFW 라이브러리는 맥 Xamarin으로 쓰기 힘들

다시 첫번째 - Python

- 개빨리 만들 수 있음
- 하루만에 만들었다
- 웬만하면 편하게 Python 쓰세요
- 게임 만들고 싶어서 들은거면 강 C++을 씹시다

두번째 - Common Lisp

- 조교를 괴롭히기 위한 최상의 방법이 무엇일까 고민
- 괄호와 괄호와 괄호

장점 / 단점

- 일반적으로 Emacs를 쓰지만 Sublime Text에 paredit을 깔면 쓸만한 에디터가 된다
- 함수형이다 - 디버깅이 편함
- 사실은 절차형으로 대부분 코딩한다
- Common Lisp가 Lisp계의 C++이라 쓸데없는 함수와 기능이 많음
- cl-opengl 라이브러리가 개발이 덜 되서 GL의 모든 함수를 지원 안함
- documentation이 전혀 안되어있음

세번째 - Common Lisp / Rust

- 김전의 꼬임에 넘어가서 Rust를 사용
- 다만 해당 시기엔 Rust가 1.0도 안나오고 GL 라이브러리도 한창 개발중이라 맥에서 왠지 빌드가 잘 안됨
- 그래서 그냥 GL은 Lisp로 그리고 Rust로는 점만 잇는걸로

Rust - 장점 / 단점

- 메모리 적으로 완벽한 프로그램을 만들 수 있음
- 컴파일러에게 코드를 던져주는 노예가 되어야 한다
- 우리가 원하는 것은 2주안에 빨리 보이는 것만 그럴듯하게 대충 만드는것
- 결론적으로 과제로 쓰는건 비추

네번째 – Common Lisp

- 했다

5번째 - Swift

- OpenGL을 안쓴다
- 좋은 언어임 (오픈GL보다 일백배)
- 코딩이 재밌어짐
- 조교가 맥이 없다
- 못씀 ㅌㅌ

5번째 - Rust

- 쓰려고 객체 디자인을 하다보니 문제가 발생
- 객체 상속이 불가능
- 뭐 안쓰고도 할 수 있는데 이거 없이 쓰는걸 생각하기 귀찮았음
- 시간도 없었음 (약 5일)

5번째 - Nim

- Ray Tracer는 속도가 개 느리기 때문에 Python 쓰기가 불가능
- Python이랑 비슷한데 컴파일러가 되서 속도가 빠른걸 찾음

The screenshot shows the Nim programming language website. At the top is the Nim logo, which consists of a yellow crown icon and the word "nim" in a stylized font. To the right of the logo are navigation links: "learn", "docs", "download", "support", "forum", and "faq". Below the navigation bar, there is a section titled "Why should I be excited?" which explains that Nim is the only language that leverages automated proof technology to perform a disjoint check for parallel code. This section includes a code snippet demonstrating a parallel loop and a comment about a compile-time error that can be fixed by changing the loop increment. To the right of this section is a yellow box with the text ". expressive", ". efficient", and ". elegant". Below this box is a "More Links" section with three links: "Community", "Aporia IDE", and "Github Repo". At the bottom of the page, there is a "Welcome to Nim" section that describes Nim as a statically typed, imperative programming language that focuses on compile-time mechanisms. Below this section is a "Latest News" section with a date "May 4, 2015" and a message that "Nim version 0.11.2 has been released!".

nim learn docs download support forum faq

Why should I be excited?
Nim is the only language that leverages automated proof technology to perform a *disjoint check* for your parallel code. Working on disjoint data means no locking is required and yet data races are impossible:

```
parallel:  
  var i = 0  
  while i <= a.high:  
    spawn f(a[i])  
    spawn f(a[i+1])  
    # ERROR: cannot prove a[i] is disjoint from a[i+1]  
    # BUT: replace 'i += 1' with 'i += 2' and the code compiles!  
    i += 1
```

. expressive
. efficient
. elegant

More Links

- Community
- Aporia IDE
- Github Repo

Welcome to Nim

Nim (formerly known as "Nimrod") is a statically typed, imperative programming language that tries to give the programmer ultimate power without compromises on runtime efficiency. This means it focuses on compile-time mechanisms in all their various forms.

Beneath a nice infix/indentation based syntax with a powerful (AST based, hygienic)

Latest News

May 4, 2015
Nim version 0.11.2 has been released!

장점

- Python-like 문법 + Static Type System
- 관측은 속도 (Rust랑 비슷하거나 약간 느림)
- C / javascript / objective-c와 같이 쓸 수 있음 (import/export 지원)
- 웬만한 C 라이브러리를 쉽게 가져다가 쓸 수 있다
- 잡다한 편리한 기능이 있음 (type inference / automatic boundary check / fast GC (non-tracing))

단점

- 개발자가 원칙이 없음 (자기가 재밌을 만한건 다 집어넣은 느낌)
- 대소문자 구분이 맨 앞글자만 / camelCase, snake_case를 같은 것으로 판단
- `light.f(x) = f(light, x)`
- `proc f(light: type_name, x: type_name): ret_type_name:`
- `method f(light: type_name, x: type_name): ret_type_name:`

그래도 쓸만한 이유

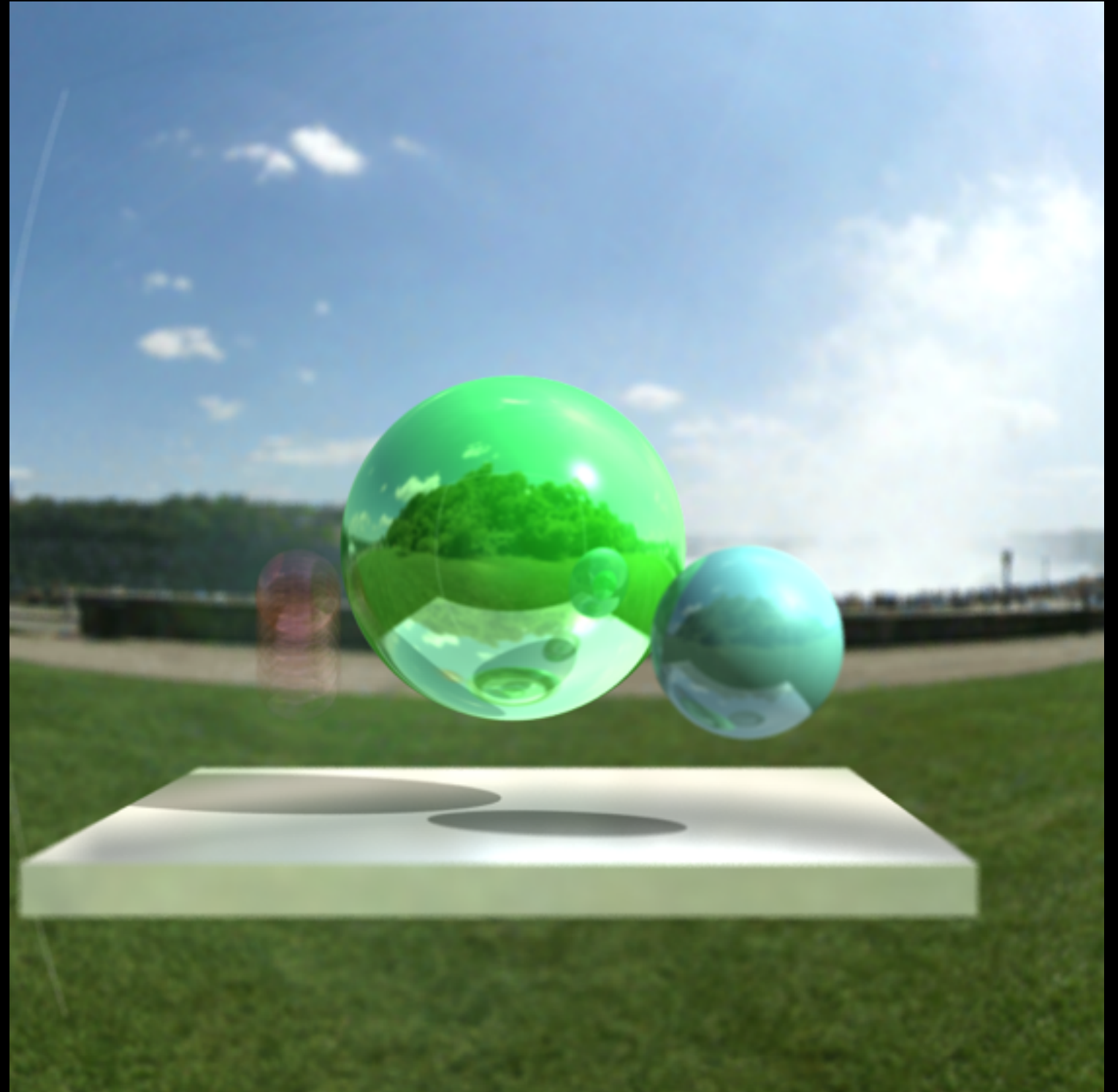
- 유지보수를 안할거면 빠르게 작성 가능
- python0이랑 비슷해서 python을 알면 속도를 보장하면서
서도 쉽게 코딩 가능
- 과제에만 쓸거라면 상당히 쓸만하다

5번 과제를 하기 위해

- 기말이라 온갖 기말 프로젝트와 시험이 산재
- 시간이 없었다
- 하라고 한거 - phong illumination, 구 그리기, 폴리곤 그리기, recursive reflection/refraction, texture mapping
- 추가구현 점수 5점씩
- 복잡한 구현보다는 추가구현 점수를 잔뜩 노리자

결과

- 좋은 점수를 받음



결론

- 내가 생각하기에 충분히 조교를 괴롭힌것 같지 않다
- 그냥 C++을 씁시다
- C++을 모르면 Python을 씁시다
- 김전을 사랑한다면 Rust를 씁시다
- 갓과목 그래픽스를 들읍시다