Rewarding code projects for *complete* beginners [closed]

Asked 16 years, 4 months ago   Modified 11 years, 4 months ago

Viewed 5k times

9

Closed. This question does not meet Stack Overflow guidelines. It is not currently accepting answers.

Questions asking us to **recommend or find a tool, library or favorite off-site resource** are off-topic for Stack Overflow as they tend to attract opinionated answers and spam. Instead, describe the problem and what has been done so far to solve it.

Closed 11 years ago.

Improve this question

Courses for people who are being introduced to programming very often include a code project, which I think is a nice way to learn. However, such projects often feel too artificial, and are thus not very rewarding to work on.

What are your ideas of rewarding code projects? (Preferably easy to begin, and extendable at will for the more advanced!).

Edit:

@Mark: thanks for the link, though I'm more interested in projects for people who are completely new to programming (the link seems to refer more to people who are already proficient in at least one language, and trying to learn a new one -the typical SO audience I'd say :) -).

@Kevin, Vaibhav, gary: I was thinking of people who are learning programming through one language, so at the beginning of the course some don't know anything about control structures (and even less about any kind of syntax). However, I was thinking in quite a large project (typically in the 1k-10k lines of code range, possibly in groups of 2 or 3 students). This is what was done at my school for the complete beginners, and it sure seemed to work for them... except that most of them found their projects quite boring to work on!

language-agnostic

Share

Improve this question

Follow

edited Aug 23, 2013 at 15:18

user1228

asked Aug 25, 2008 at 20:21

OysterD

6,750 ● 5 ● 35 ● 33

7 Answers

Sorted by: Highest score (default)

4

As has been stated a few times, what you are trying to teach the beginner is very important to the project.

My advice to you for planning something like this:

1) Avoid making a computer game

A computer game, while fun to build, doesn't reward the programmer with results early on (it's very complex). You want to concentrate on small but useful application programs, such as a Port Scanner. The example there is a little complex, but it's one of the best learning projects I've seen on the web.

2) Teach graphics early

It's rewarding to see the fruits of your labors early on, and it motivates you to go further. Whether you're using WinForms, MFC or the Win32 API, OpenGL or DirectX, teach it early.

3) Many small lessons with in depth information

This principle is followed by the above linked Port Scanner project, and it works well. Teach each part thoroughly, and give time for the beginner to absorb the lesson. I think that ZophusX had a good format for giving the information. It's too bad he's mostly abandoned his site.

4) It takes time

Don't rush things. Nobody becomes a stellar programmer in a few weeks. Try and make the lessons simple, but engaging, and keep building from your previous lessons.

5) Get feedback early and often

You might think a project is incredibly interesting, or a particular lesson or such, but you aren't the one learning. Your student(s) will greatly appreciate it when you ask them early on how things are going, and what they'd like to know more about. Be flexible enough that you can accomodate some of those requests.

6) Have fun teaching

Have fun. Passion is contagious, and if your student(s) see how much you enjoy the subject matter, some of that enthusiasm will rub off on them as well.

I hope that helps!

Share Improve this answer

Follow

answered Aug 25, 2008 at 20:47

Marc Reside

2,956 ● 1 ● 24 ● 26

The link to the port scanner code seems to be broken on that site. By any chance do you still have a copy? I contacted the site but they didn't reply. – Steve Aug 16, 2009 at 18:22

2

Some good rewarding projects, in terms of what you can learn and which are quite scalable in terms of complexity, features are:

- Games
- A travel and transportation reservation/booking system
- Encyclopedia or a Dictionary of terms, articles
- Conversion Calculators (Currency, Units, etc.)

The key is to pick a project simple enough, so that some of its features are immediately apparent, when you look at the project title. And when really given a thought, will reveal more features that you can add to it.

The project should have enough difficulty to so that its features seem just beyond the beginner's reach, thereby motivating him to learn something new all the time.

Share Improve this answer Follow

answered Aug 25, 2008 at 20:42

Pascal
4,127 ● 8 ● 34 ● 29

1

If you are training new people in your company, then attaching them as intern resources on a live project is very rewarding.

This increases the work load of the main developers a little (because they have to review all the work that the intern does), but goes a long way in terms of training and development of the person.

Share Improve this answer Follow

answered Aug 25, 2008 at 20:32

Vaibhav
11.4k ● 11 ● 53 ● 71

1

I do think that games and puzzles are a good place to start as they can give great scope for developing more complex versions. For example a **tic-tac-toe** program can be built as a simple command line program initially that lets two players play the game.

This step can be used to show how a simple data structure or array can represent the game board, simple input to get user commands/moves, simple output to display the game board and prompts etc. Then you can start showing how an algorithm can be used to allow player vs computer mode. I like the simple magic square math algorithm for **tic-tac-toe** as it's based on *very* simple math. After this the sky's the limit, UI improvements, using file I/O to load and save games, more advanced algorithms to get the computer to play better etc. More complex and satisfying games can still be produced using text mode or simple graphics.

I've used the **Sokoban** game as a means of showing lots of techniques over the years.

The simplest game I've used is a number list reversing game. This involves a mixed up list of numbers from 1-9.

The player can specify a number of digits to reverse on the left of the list. The aim is to get the list sorted. This is great for absolute beginners. Each little part of the game can be written and tested separately.

answered Aug 29, 2008 at 23:01
Tim Ring
1,833 ● 1 ● 20 ● 27

0

It really depends on what you're trying to teach the beginner. If you're trying to teach syntax, then simple "Hello World" programs and ones that spit out every odd number between 1 and 100 are fine to get them started. If you're trying to teach data structures, then maybe something like a 20 questions game or some simple sorting program. If you're trying to teach recursion, then maybe a breadth first search program. If you're trying to teach database manipulation, then something like a order tracking system would be appropriate.

answered Aug 25, 2008 at 20:26
Kevin Pang
41.4k ● 38 ● 122 ● 173

0

Take a look at code examples in the book [Python Programming for the Absolute Beginner]()

answered Jan 7, 2009 at 13:08
jfs
413k ● 205 ● 1k ● 1.7k

0

Text Adventure.

- It's a console app
- You'll need to do some useful things, hold inventory, map and room state and parse input
- It's fun and you can give it to others to play! :D

Share Improve this answer

Follow

answered Jan 7, 2009 at 13:27

Quibblesome

25.4k ● 10 ● 62 ● 104