Kid's Programming Language (KPL)

Jon Schwartz, Jonah Stagner, Walt Morrison
Morrison Schwartz, Inc.

ABSTRACT

In this paper, we introduce Kid's Programming Language, or KPL. KPL is an integrated development environment (IDE) and programming language which are similar to but greatly simplified from current mainstream IDEs and languages. KPL is educational freeware. KPL was initially targeted at the 10-14 age group, but has proven to be engaging and interesting to beginning programmers and hobbyists of all ages. KPL offers a highly leveraged object model which emphasizes graphics programming, including 2D and 3D graphics. KPL intends to address the problem of declining computer science interest and enrollment by 1) making it easy for beginners to get started with computer programming, 2) capturing and holding beginners' interest by emphasizing graphics and games programming and 3) enabling a smooth "graduation" from KPL into mainstream languages and IDEs.

CATEGORIES

D.3 [**Programming Languages**]. I.3.6 [**Computer Graphics Interaction Techniques**]. K.3 [**Computers and Education**]. K.8 [**Personal Computing: Games**].

KEYWORDS

Computer Science education, beginning programming, 3D graphics, 2D graphics, games, programming language, integrated development environment

1. INTRODUCTION

The current crisis in Computer Science enrollment is bringing renewed attention to the need for and importance of programming languages and environments which are usable by and interesting to beginning programmers. The percentage of 2004's incoming US undergraduates who indicated they would probably major in Computer Science declined by over 60% from a peak in 2000, just four years earlier. The decline in interest among women was even steeper, with the number of women indicating interest in Computer Science dropping to levels not seen since the early 1970s.[1] These declines are ironic given the statistics about computer and video game software sales, which continue to grow, and which more than doubled between 1996 and 2005.[2] Also significantly, 43% of all computer and video game players are women[3], and 69% of all adult women use the Internet[4]. Clearly the lack of undergraduate interest is much more specific to Computer Science as a profession than it is to general usage of computer technology.

There are many factors for us to consider about this crisis, but in introducing KPL's relevance to this situation we will focus on three high level goals which KPL is designed to address. You will note that only one of these goals is technical. Another is social, and the third is about individual creativity. The current crisis is computer science is not solely a technical problem, and it will not be solved with a solely technical solution.

1.1 Reduce Technical Barriers to Entry

Computer Science as a field of study and technological profession continues to advance extremely rapidly. Consider for a moment the long list of programming languages, supporting technologies and software development tools and techniques which have evolved in only the past decade. This rapid progress is a key to the continued growth in software applications that improve the way the world lives, works, communicates and does business – and so this rapid progress is not likely to slow. The problem which we now face is that while we have pushed Computer Science so rapidly forward, we have spent **very** little time considering the beginning programmer. As a result, the learning curve for a beginning programmer has become painfully and discouragingly steep.

This defines KPL's most important opportunity and most important goal: to reduce that learning curve as much as is possible, and to reduce the technical barriers to entry that beginning programmers must face. It is not sufficient to consider only the beginner scenario, however, since the goal is not **only** to enable and interest beginners. It is also important to encourage and prepare beginners to move on to the more powerful mainstream programming languages and technologies which they will use in academia or as professionals.

1.2 Leverage Social Interest in Computing

Social interest in computing, in our definition, is the extent to which use of computing devices becomes an active part of our societies and our lifestyles. Social interest in computing is becoming the norm, not the exception, and is increasingly understood to be the engine which is driving technological advancement. Social interest in computing is particularly clear if we consider the convergence which is happening around mobile computing. Cell phones, portable gaming devices and media players, for instance, are converging toward near-future mobile devices which will offer all these functions, as well as those of a fully capable and fully programmable computer.

There are two ways that KPL intends to leverage social interest in computing. First, the best way to interest users in programming is to enable them to program the kinds of applications that interest them. On this point, KPL first emphasizes applications based on graphics, and in particular, graphical games. KPL's support for graphics, sound and game development are complete enough that, even though it is a beginner's environment, it is fully capable of producing commercial-quality casual games. This is indeed proving to be one of the most important features that draw beginners to KPL, and keeps them working in KPL. Extensions to the KPL libraries and object models are planned specifically based on social interest in computing as confirmed by numerous requests from KPL users. These plans include 1) an abstraction

layer making it easy to implement "chat" or "IM" applications using KPL, and 2) an abstraction layer adding network communications to KPL in support of multiplayer games.

The second important way KPL will leverage social interest in computing is to allow users to use and share their own applications on the platforms that fit their own social network and lifestyle. KPL can currently produce executables that will run on Windows computers, which covers more than 80% of the world's PCs. The importance and value of cross-platform support is obvious, however, and KPL programs should in the future be able to run on other platforms, including browser-based, the Java Virtual Machine, various game consoles, other operating systems, and on cell-phones.

1.3 Enable Creative Self-Expression

This point wanders far from academic computer science thinking, but even so there are at least two ways it is worth our consideration. The first is simply about engaging beginning programmers. No matter the age they begin, we must hold the beginner's interest long enough for them to develop a basic understanding of what computer programming is all about. Only when they have reached that point can they make an informed decision that they would like to pursue Computer Science further.

We have designed and built KPL based on the belief that there are many **non-programmers** who might find a career in Computer Science as fun and satisfying as we ourselves have. We were all once non-programmers. Whatever a beginner's particular interests are as they begin, allowing them to program applications or games based on these interests is an obvious way to engage them. Open source KPL examples contributed by users to the community include user-created animated cartoons, dynamic graphical art of various kinds, custom images and models, interactive fiction, and user-recorded voice and sound clips.

This point is also important because content in the form of 3D models, digital art and images, sound, music, icons, game or UI text, etc… is increasingly important in the production of software. Big budget games already spend far more time and money on such content than they do on game code. Computer Science programs which are leading the way in offering game development courses as part of their curriculum demonstrate the importance of this point by making those courses a joint offering with a Visual Arts or Design program.[5] Whether a beginner chooses to move forward with a focus on content or on coding, KPL offers them an experience and understanding of both. Collaboration on technical **and** creative aspects of software development also offers a useful experience of cross-functional teamwork.

2. KPL FEATURES

KPL should be considered a fully capable and very modern software development environment, albeit a simplified one.

2.1 LANGUAGE SYNTAX

KPL's language syntax is, most importantly, readable, and because of this it is most similar to BASIC. It is, however, a structured language as much as is C# or Java. Supported data types include Integer, Boolean, String and Decimal. User-defined functions and methods are supported. User-defined classes are supported as well – though not inheritance or polymorphism. KPL's support for classes is intended to allow experience and use of a component-oriented programming model, not a full object-oriented programming model:

```
Define fps As TextLabel
fps.MoveTo( 10, 0 )
fps.Color = Colors.Yellow
fps.Text = "Loading terrain"
```

2.2 CLASS LIBRARIES

Extensible class libraries make many classes, methods and functions available to the KPL programmer, also carefully designed and named for readability. These class libraries are designed primarily to enable programming of games, graphics and sounds. Another very important design point is that they are highly leveraged: they encapsulate as much complexity as possible, to allow the programs that use them to be as simple as possible. As an example, a 3D model or a 2D sprite can be rendered and moved with 3 lines of KPL code:

```
Define pointer As Model3D
pointer = LoadModel( "Arrow-yel.x" )
pointer.MoveTo( 0, 0, 0 )
```

The KPL class library architecture is extensible, and an SDK is planned. The current list of libraries is: Collections, Colors, ConsoleMode, DateTime, Drawing, FileIO, Fonts, Game3D, GameInput, GameUI, Keyboard, Keys, Math, Mouse, Shell, Sounds, Sprites, Strings, SystemColors and Timers.

2.3 CODE EDITOR

KPL offers a code editor modeled after Visual Studio.NET and Eclipse, and which offers many of the discoverability, usability and productivity enhancements available in those mainstream environments, including syntax color-coding, Intellisense, Autocomplete, mouse-hover tooltips and collapsible code regions. Compiler warnings or errors are visually indicated directly in the code editor, and the details of the warning or error is available with a mouse hover over the offending line of code.

The pedagogical advantages of these features are clear: the code editor itself teaches a beginner how to program as they type. As a beginner progresses, these features offer the same usability and productivity advantages in KPL as they do in Eclipse or Visual Studio.NET. And, of course, they prepare the programmer to graduate to these or other mainstream IDEs.

2.4 DEBUGGING IN KPL

Code debugging is a critical part of learning the skill of computer programming, and KPL offers simple and standard support for debugging. From a breakpoint, the KPL user can Run, Stop, Step Over, Step Into or Step Out. When a user is stopped at a breakpoint, a dockable Debug window is available which lists all in-scope variables, their data type, and their current value. Those of us who are now experienced professionals may find it hard to appreciate the pedagogical value of interactive debugging. Consider the beginner scenario for a moment: there is no better way to learn looping, logic, or function and method calls, than interactively stepping through them one instruction at a time. Similarly, there is no better way to understand variables and scoping than to observe as they move in and out of scope, and as their values are updated by KPL instructions.

2.5 IDE MENUS AND TOOLBAR

KPL's menus and toolbar are effectively a subset of those available in Visual Studio.NET or Eclipse, and as such are less daunting for a beginner, as well as good preparation for graduation into those or other mainstream IDEs.

2.6 IDE TOOL WINDOWS

KPL's IDE offers standard support for a simplified set of IDE tool windows. Tool windows can be floating or docked, and when docked can be pinned open or collapsible. The functionality they offer can be optional based on the user's experience and interests, and include:

Messages (for compiler or other system messages), **File Explorer**, **Program Explorer** (a hierarchical and navigable tree view outlining the KPL code), **Notes**, **Debug** (all in-scope variables, their type and their value are automatically displayed when at a debug breakpoint), **Framework Browser** (tree view of the loaded class libraries, and the classes, methods and functions they make available), **Picture browser**, **C# Conversion** and **VB.NET Conversion** (because KPL is a .NET language, it is easy to convert KPL code to other .NET languages – other languages should be supported in the future).

3. KPL DESIGN POINTS

Our introduction presented our high level goals for KPL, but we have found it useful to define a dozen specific lower-level design points. On the one hand, we sometimes state the obvious with these – but on the other hand, these points have **not** been universally satisfied by academic Computer Science programs or by other programming tools.

3.1 FUN

"Learning is best when learning is fun" has been our KPL motto. This is also a critical point in response to the Computer Science enrollment crisis. "Fun" is clearly an individual and subjective perception encompassing many factors – but it is constantly in our minds as we work on KPL, and on supporting content and materials.

3.2 ACCESSIBLE AND USABLE

Accessibility determines whether a user will ever actually get started – with KPL or with anything else, including Computer Science programs. Usability issues are critical for user retention as well as pedagogy.

3.3 ENGAGING

Graphics, games and sounds are highly interesting and engaging, particularly for a typical beginner demographic from 8 to 18 years of age. Note that KPL offers this engaging focus while still being a complete, flexible and functional programming language and IDE. KPL content **is** engaging, but it is **also** Computer Science.

3.4 SIMPLE

A pilot could begin learning to fly while sitting in a 747 cockpit, and a programmer could begin learning to program while sitting in front of Eclipse or Visual Studio.NET. We built KPL as we did because we believe learning **should** begin in a much simpler and more comfortable way.

3.5 REWARDING

Graphics and sounds are ideal rewards to present to a beginning programmer as immediate, vivid and satisfying results of their work. Graphical examples are the basis of the Beginning Programming Tutorial which we have provided for use with KPL.

3.6 HIGHLY LEVERAGED

Maximum function from minimum code is gratifying to programmers of any skill level – but when we are trying to interest a beginner in Computer Science, it is even more critical. This goal is accomplished in KPL by hiding and implementing as much coding complexity as possible within a well-designed class library.

3.7 PROGRESSIVE

An absolute beginner requires an environment and experience which allows them to successfully start, despite a background of no programming knowledge. Very quickly, though, an absolute beginner advances, and is ready to learn more complex concepts, algorithms and tools. The programming environment they begin in should offer considerable progression beyond the absolute beginner experience. KPL does this in numerous ways, offering optional tools in the IDE, and offering advanced concepts that a beginner can grow into, including but not limited to: 3D graphics coordinate systems, recursion, user-defined classes, and building and distributing complete and playable casual games.

3.8 PREPARATORY

The ultimate goal of any beginner environment must be to prepare a beginner to move on to other languages and IDEs when they are no longer a beginner. It is clearly a delicate design balance to provide a simple experience for a beginning programmer, while also preparing them as much as possible for the languages and tools they will use in their future academic or professional work.

3.9 MODERN

Consistency with current software design standards is an important usability point. The more the look, feel, commands, menus and functions in a new environment are like the software the user already has experience with, the more quickly and easily they will learn that new environment. Environments which are radically different from common current software design standards can also suffer from a negative reaction simply due to these differences, and their resulting unfamiliarity.

3.10 PUBLISHABLE

For many programmers, satisfaction that others are happily using programs we wrote is the best reward and encouragement that comes from our programming work. Even as a beginner's programming environment, KPL offers this satisfaction in two ways. A simple KPL program is contained in a single text file, and these files can be freely shared, posted, emailed, loaded and run in any user's copy of KPL. KPL also builds EXE files of each KPL program as it runs them, and these EXE files can be distributed and run on other computers, including those which do not have KPL installed.

3.11 STATE OF THE ART

Software standards and technologies evolve extremely rapidly, and as a result, the useful lifetime of a particular piece of software can be limited by the viability and lifetime of the standards and technologies it is based on. KPL is based on the .NET Framework 2.0, and the latest design and technology standards of Visual Studio 2005 – each of which released in November 2005.

3.12 INTERNATIONAL

KPL utilizes an open strings architecture which has allowed volunteers to translate the KPL IDE for use in Russian, Chinese, German, French, Spanish, Portuguese, Italian, Dutch, Danish, Swedish, Polish, Czech, Greek, Romanian and Catalan. All those translations were completed within the first seven months after KPL's release, and additional translations are in progress.

4. COMPARING KPL TO OTHER LANGUAGES

We have ourselves used the design points above to plan and to evaluate KPL. We encourage your own evaluation of KPL, using these points or your own criteria. Clearly, the relative importance of the points above will vary based on where and how KPL is used.

We have often been asked to compare KPL against other beginning programming languages which are already available and in use, such as Logo, Squeak and Alice. We appreciate each of these languages, and all of the work and effort that has gone into them as they also help and enable beginning programmers. Our own assessment is that Logo, Squeak and Alice each focus on and deliver well on the first six design points listed above: fun, accessible and usable, engaging, simple, rewarding and highly leveraged. Indeed, they may have advantages over KPL on certain of these design points, for certain user scenarios.

However, we believe KPL offers significant advantages over these or any other beginning programming environment based on the remaining six design goals listed above: progressive, preparatory, modern, publishable, state of the art and international.

We expect the CS enrollment crisis and the resulting focus on the need for good beginning programming environments to mean that KPL, Logo, Squeak and Alice will each receive increased priority, resources and development effort over the next few years. We expect a certain amount of cross-fertilization to happen across these environments, as the best features of each make their way into the other environments as well. This will clearly be a good thing for the languages as well as for users.

5. GLOBAL KPL COMMUNITY

KPL began as and remains educational freeware. As such, KPL has never had more than word-of-mouth and word-of-blog marketing, and to date has had little media coverage or academic exposure. Despite these facts, KPL was downloaded 55,000 times in the seven months after its release in July 2005. User response to KPL is generally enthusiastic, and as KPL awareness and visibility grows, many more people are discovering it every day.

KPL is a software product created, supported, translated and maintained by a steadily growing and global community of volunteers. In the first seven months after KPL's release, volunteers around the world translated the KPL IDE into 15 languages beyond the English default, and several other translations are in progress. KPL is used by schools and individuals around the world, in local native languages, thanks to these volunteer translators.

Volunteers have also translated the entire KPL website into Spanish and Portuguese, with other site translations also in progress. Similarly, a beginning programming tutorial written for use with KPL has been translated into three additional languages thus far, with additional translations in progress. Translation of the website has turned out to be doubly important: not only does the translated site better support readers in their own languages, but the indexed translation of the site also brings visitors to the site based on web searches done in their own language.

Volunteers have also contributed dozens of open source KPL programs to the community. These programs range from simple educational examples written for beginners to fully functional games. As educational and open source, these examples are consistently carefully written and carefully commented.

6. CONCLUSIONS

The usefulness, applicability and value of KPL for beginning programmers are already demonstrated by enthusiastic and successful use in classrooms and homeschools around the world.

The most immediate requirement which will allow increased academic adoption of KPL is for curriculum materials which support that academic adoption. Volunteers are working on these curriculum materials, and individual teachers are producing materials for use by their own classes.

A longer-term requirement for moving KPL forward involves broadening KPL to other platforms and technologies, and deepening it with additional class libraries based on specific application or education goals. Examples of these efforts range from small focused efforts (libraries specific to a particular university course, for instance) to efforts the size of KPL itself (for example, porting KPL to the Java VM).

Institutions considering large-scale use of KPL are also asking obvious questions: Where will KPL be in five years? How will it be supported?

Each of these requirements demonstrates the need for us to define a sustainable model for moving KPL forward over time. Placing KPL into the open source community is one option we are considering, but we do not believe this is necessarily the best solution. Forming a non-profit around KPL is another solution. Commercializing a version of KPL while continuing to offer an educational freeware version is also a possibility. The approach which makes the most sense to us, given the immediacy of the enrollment crisis and the fact that KPL is ready now to help address it, involves transferring KPL into a larger academic or corporate institution which has the resources, visibility and connections to accelerate KPL's global success and impact. We are evaluating each of these possibilities, and welcome suggestions about them.

KPL is in use around the world today, and is already addressing the problem of bringing beginners to Computer Science. We look forward to seeing the KPL community grow further around this fun and exciting project. And we look forward to seeing where that community will take KPL.

7. REFERENCES

[1] Vegso, Jay. Interest in CS as a Major Drops Among Incoming Freshmen. Computing Research News, Vol. 17/No. 3, May 2005

[2] Entertainment Software Association, 2005 Sales, Demographics and Usage Data. http://www.theesa.com/files/2005EssentialFacts.pdf

[3] Entertainment Software Association, 2005 Sales, Demographics and Usage Data. http://www.theesa.com/files/2005EssentialFacts.pdf

[4] Pew Internet and American Life project, Demographics of Internet Users, December 5, 2005.

[5] Parberry, Ian, et al, The Art and Science of Game Programming, SIGCSE '06, March 2006. http://www.eng.unt.edu/~ian/pubs/fp107-parberry.pdf