Towards a Game-based Programming Learning Environment for Kids

Ralf Schmidt, Maic Masuch, Julia Othlinghaus

Entertainment Computing Group, Faculty of Engineering, University Duisburg-Essen

Abstract

In this paper we describe our vision of an engaging, game-based programming learning environment for novice learners aged 10 to 13. We aim to support class teaching situations until pupils reach an advanced level of knowledge, allowing for a smooth transition to general purpose languages. As a first hands-on approach, the prototype "Plumps" is presented, which features scenarios with goal-based didactics as well as selected aspects of a comprehensive game-based motivational concept. We further discuss the positive and constructive feedback of an informal evaluation carried out among 21 secondary school children to assess the general acceptance. Finally, we conclude with a reflection and discuss next steps towards a further multidisciplinary research and development.

1 Introduction

Information technology and digital media tend to accompany and dominate our everyday life with an increasing level of speed. It is therefore important to develop a solid body of information-technology literacy across all ages to foster a responsible and beneficial use of technology as well as enabling people to participate in the development. In fact, for today`s kids, computer literacy is crucial. Accessing, processing, and evaluating information has become a basic qualification and common knowledge (Horz 2004, Reichert 2001) and also grown to an important success factor for school and later work-life. But only small numbers actually acquire a deeper knowledge beyond the use of common applications, like word processors for homework, (Krotz & Hasebrink 2003) or leisure activities like chat, browsing or gaming.

To learn the fundamentals of software development is probably the best way of acquiring knowledge about the internal processes of a computer and principles of information processing through software. It also enables creating solutions for specific problems in commercial (e.g. mobile apps) or non commercial contexts, which can be highly satisfying, motivating and even sometimes economically rewarding. Furthermore, an increasing knowledge about software development is accompanied by related important competences, such as mathematics, logical- and problem orientated thinking (Kurland 1984).

In this paper we describe our vision of and first steps towards an engaging game-based programming learning environment (PLE) for young children aged 10 to 13 years, with little or no prior knowledge in this field. We especially aim at supporting class teaching situations. The task demands strong collaboration of several disciplines such as computer science and HCI as well as didactics and psychology. Preliminary to setting up such a multidisciplinary team, we developed a first prototype to create a solid base for discussion and further research. Our overall goal is to create an environment that allows learners an easy change-over to later programming in general purpose languages (GPL) in the long term.

2 Related Work

A number of PLEs for beginners and especially children have been realized over the past decades. Among related PLEs, such as Greenfoot, Squeak Etoys or BlueJ, we investigate four representatives in a little more detail as they are more strongly related to our approach. The PLEs are classified into five different categories defined by (Powers et al. 2006). For a comprehensive overview and a more detailed categorization of PLE refer to (Kelleher & Pausch 2005).

Visual programming tools aim to avoid textual syntax problems, by only allowing certain combinations of visual programming building blocks, determined by their shape and color and an overall reduced complexity. *Scratch* is a typical and very successful example. Resnick et al. (2009) name the three exemplary main principles of the software: A "Lego-block" approach to programming where false combinations are avoided by the shapes of the blocks. Moreover it is meaningful to its users by easy personalization attributes and a tight connection to the Scratch website, which features social activities. The goal of scratch is explicitly not to prepare kids for GPLs. Resnick et al. state that with Scratch, the MIT Life Long Kindergarten Group will keep a low level and wide approach and suggests that children with a deeper interest in programming need to move on to other languages.

Boles' *Java Hamster Modell* (Boles 2008) is a typical *tiered language approach* with elements from visual and narrative tools. It represents one example of a number of environments. Learners use a given small set of commands (mini language) to control some kind of 2D-character in a squared game board (mini-world). Typical tasks consist of pick-up and drop actions, combined with movement and evasion of obstacles. This kind of tools still represent a high abstraction level compared to GPLs but require the actual typing of program code along with a basic insight of syntax and semantic as a special characteristic. Therefore, depending on the tasks given, the learning curve can be steep. The motivational factors for beginners are low beside the visualization of the game board because the UI is relatively complex and the often quite technical feedback hard to understand. However, when acquainted with the environment, the Java Hamster Modell reveals itself as a powerful and extendable tool.

Kara was developed by ETH Zürich to allow learners an easy start by using elements from visual programming. It is an example for *flow model tools*, which focus on visualization of

algorithms and program flow. Programs are created by combining commands and adding attributes which are then represented by a ladybug in a 2D playground. The visualization of program flow as finite state machines supports comprehension. With Java Kara the developers offer advanced learners to change-over to actual object orientated Java programming while still visualizing a programs behavior using ladybug and playground. The idea to keep learners in their known environment with visual representation offers a relatively low-contrast while giving an idea of the power of GPLs.

Lego Mindstorms© represents a class of tools that embodies behavior and states of a program in tangible real life artifacts. With Mindstorms©, a construction block kit containing programmable motors and sensors, which can be combined with other non-interactive blocks, allows to construct a wide variety of objects. To control the programmable buildings blocks, learners use a corresponding PLE. The integration and control of artificial objects is considered to be extremely motivating to kids and fundamentally reflects the idea of Papert`s "objects-to-think-with" (Papert 1993). Lego Mindstorms© is used in schools, but only in quite structured and prepared lessons as the system is quite complex for beginners (Powers et al. 2006).

Narrative tools are authoring environments that use programming to visualize stories or parts of stories. To create and tell the stories is, besides the aesthetical aspect the main motivational factor. One well known example of a narrative tool is *Alice 3D*, developed by Pausch et al. at Carnegie Mellon University.

In the next chapters we briefly present an overview of our theoretical base and conclusions for the prototype concept.

2.1 Visual vs. Tiered Language Tools

Our overall goal is to create an engaging environment that prepares users adequately for later coding in more powerful GPLs. Therefore, a discussion about the optimal category or category mix is necessary. The most controversial topic is about a visual vs. text-based approach. Research done on the topic, such as by Schiffer (1996) and Hundhausen et al. (2002), lead to the assumptions that at a beginner's level visual programming tools, with their more appealing look and feel and (failsafe) drag and drop interfaces, can help to avoid early frustration in the field of programming compared to a more complex text syntax and code editors. A text based approach would need a strong motivational concept and an easy-entry design to compensate. On the other hand, with a visual approach, the complexity and depth of a programming task is relatively limited compared to text based approaches because of the smaller information density. Overview is lost quickly in larger visual code representations. More research is needed towards the question which mental model, build up during the interaction with the tool-types, is easier to transform to actual GPL programming. For example, (Hundhausen et al. 2002) deprived any additional value by the mere visualization of algorithms alone. The feature would only support students if constructed in terms of constructionism.

2.2 Didactics and Motivational Aspects

The fundamental ideas of constructionism were chosen as an underlying concept for our approach. The theory considers learners to be *active constructers* of knowledge within an open and flexible environment. By the creation of such real or virtual *objects,* that are ideally *meaningful* to the learner, the process of learning can be rendered more successful (Papert 1993). Based on constructionism and corresponding didactical concepts, media didactics include further theories on how the conception of media-supported learning could be realized (Kerres 2001). Klein emphasizes, that a sound didactic design is crucial for the success of a learning environment and should be backed up by a deep knowledge about the target group and context as well as clear defined goals (Klein 2000).

That knowledge is also crucial for the "motivational design", by which an environment should offer an appealing easy-entry as well as long term motivational factors. In general, children perceive the process of learning itself as fun. They naturally seek challenges and eagerly adopt new knowledge. Learning environments of any kind should seek to support that intrinsic motivation and natural eagerness as much as possible. They should respect the learners need for autonomy, encourage to experiment and provide chances to broaden the knowledge by offering different approaches to tasks with slowly increasing challenges. That progress should be made visible by short and positive feedback cycles (Grubert 2009).

Another way to foster motivation is to personalize an environment by allowing adaptation of the environment and its content. Where applicable, user generated content is a promising way to individualize an environment autonomously, which often renders it more meaningful to the user. In comparison, many relationships between the abovementioned opportunities and goals of learning environments and the principles and mechanics of games can be found (Wölke 2009). Charsky (2010) examined these similarities by drawing connections between the characteristics of constructivism learning theory and those of serious games, which can be considered as a special form of game-based learning environments. For Charsky, goal-based scenarios consist of seven relevant components: learning objectives, cover story, mission, scenarios, role, resources and feedback. Those characteristics, as Charsky states, can also be found in serious games or game-based learning environments. For example, the learner's current task can be defined by a given scenario and linked to goals that support the overall mission and fit to the cover-story. Whether goals, resources and feedback should be carried out by an instructor or not is in dispute. However, if designed carefully, the instructor's role may also be covered and audited by one or more non-player characters. We extend this notion by assuming that – to a certain degree – in a game-based learning environment, this role also can be fulfilled by a carefully designed task and feedback system.

2.3 Target Group

For our PLE the target group is children between 10 and 13 years of age as cognitive research shows that deeper and more complex knowledge evolves not until 11 years of age (Schwill 2001). Typically, those pupils attend school classes five to seven in western countries and usually do have a relatively broad access to computers (Feierabend et al. 2010).

Also, the use of a computer is highly ranked in the domain of interest (most kids would miss it the second-most right after TV). When asked about, communicating, browsing the internet and gaming are the most referred activities. These results might give a hint why the popularity of the computer is tightly bound to the notion of fun. However, regarding computer literacy, findings suggest that the target group possesses only very limited knowledge about how computers actually work. However, they had a positive interest in programming, even when most of them could not explain the term (Sheehan 2005).

2.4 Classroom Context

With our PLE, we aim to support classroom teaching but do not abandon the concept of individual learning at an own pace. For the use in classrooms, an appropriate IT infrastructure is needed. With D21, a long term technology initiative of the German government, approx. 90% of German schools can offer their pupils a general access to computers. Also, with 80%, most teachers favor the use of computers in classrooms but are in the need of support regarding how to integrate the technology easily and effectively in class (TNS Infratest 2011). Considering computer science as school subject however, more experienced teachers and computer classrooms are common. Therefore, a lot of schools offer their pupils basic text-based programming (as an elective course) and preparation for the ECDL (European Computer Driving License) certificate. Still, in Germany teachers only seldom integrate game-based approaches into their lessons as the use of games or serious games is not widely accepted and a matter of intense discussion. A number of initiatives contribute to the discussion by offering information and ideas for teachers about the use of computer games in schools (Felicia 2009, LfM NRW 2010).

3 Concept

Taking the aforementioned findings into account, we decided to pursue a tired language approach, enhanced by visual elements, a clear and supportive GUI, smart coding features and a sound motivational design. Regarding didactics we aim to support goal-based scenarios in general, but stay open towards other concepts, such as problem-based learning, which would demand group features. Our first steps towards such an environment concentrated on the following main goals:

1. Children should gain a basic understanding of programming as such, through a simplified tiered language approach.
2. First implementations of game elements should support the learner`s motivation and provide an easy entry.
3. The environment in principle should be highly scalable to reflect the learner’s abilities.
4. The basic didactical concept is based on constructionist learning theory principles and follows a goal-based scenario approach.

The prototype should be used to gather first feedback from the target group by conducting informal interview sessions. To quickly gain first results, we used *Solist*, a development framework to create micro world PLEs similar to the Java Hamster Modell. The framework is Java-based and aims to allow e.g. teachers without a deeper knowledge in the Java-JDK to create their own micro worlds, so called theater plays. Solist consists of a graphical IDE, a fixed Java class library and the so called Theater-API, which allows the individual creation of micro worlds within fixed bounds. Because of its predefined structure, the use of Solist allowed us to implement a prototype including assets and a first lesson in less than one man month. On the other hand, many compromises had to be made as the API only offers few possibilities to adapt the predefined structure, especially the UI and interaction design.

3.1 Didactical & Motivational Concept

For our prototype Plumps we implemented a number of goal-based scenarios, corresponding to suggestions by Charsky (2010):

Learning objectives, structure of learning material & scenarios: Plumps provides eight consecutive learning sections like first steps in controlling the player character or the creation of individual procedures and conditional statements. For each section it allocates learning material like instructions, explanations and examples to give an idea of the topic and how to use it. Up to three exercises, accompanied by a concrete definition of tasks, scenarios and hints about how to find the solution provides the novice learner with structure and guidance throughout the exercises. All tasks can be solved by using a visual command window or actual writing code. A very simple mini-language was developed, with commands related to the environment and cover story. By its given features, the Solist environment also allows more experienced learners (or the instructor) to construct their own levels and goals from scratch and solve them with friends by using the editor.

Cover story and role: As we aim for pupils from 10 to 13 years of age, a very simple story was developed and represented in the graphical assets of the environment. As a synopsis, an evil professor has kidnapped Mimi the mouse and her friends to conduct unpleasant experiments with them. Now Mimi and the others have to find and collect magic stars in the professor's house, which were arranged by a fairy, to gain freedom. With each level, the learner can rescue one player character by solving the given tasks.

Resources: There are two in-game resources the player has to take care of. First, a specific score needs to be reached by collecting assets within a level, representing the food for the player character. Furthermore, the number of lines of code a learner is allowed to write is restricted in order to motivate the usage of control structures.

Feedback & motivational factors: By default, the Solist environment allows a direct visual feedback of written code through the virtual playground. An alternative step-by-step program execution allows comprehending the connection between written commands and the reactions of the player character in detail. With the score for feeding food-assets as well as win/lose messages, learners get further, very simple feedback to their progress. We did not implement a more complex feedback system for the first prototype as we rather concentrated

suppressing and changing the standard feedback messages of Solist, as these are not suitable for the novice learner.

3.2 GUI Concept

The Graphical user interface of Plumps is rather basic as the concept of the Solist framework does not feature changes to the default look. The GUI consists of two main areas: A source code editor and a simulation area (playing field). Children can use the editor to write programs and view results in the simulation area. The introductory tutorial, learning material and exercises are realized as separate windows that mainly contain text and images. Tutorials, tasks and feedback are also represented in extra windows.

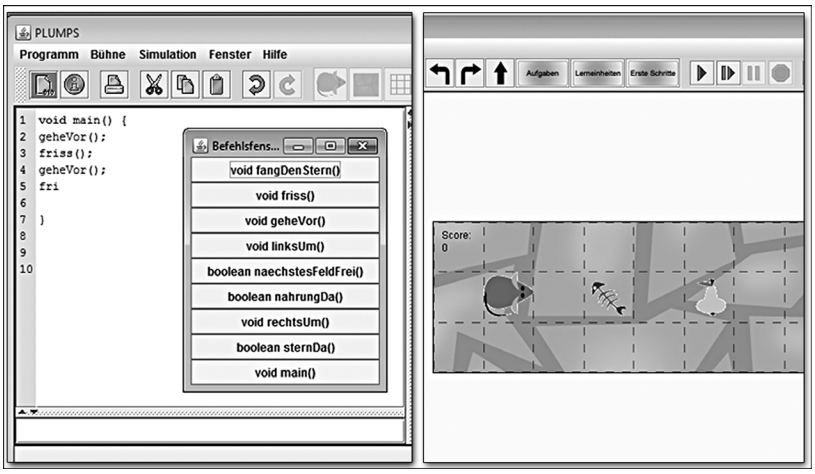

Figure 1: Code editor, visual command window and virtual playground

3.3 Informal Target Group Interviews

With regard to a user centered approach, we gathered feedback from the target group during two informal interview sessions. Our goal was to evaluate our target group information and to get early feedback about the general acceptance of the environment. With the representation of some simple tasks we tried to get an idea about the intensity of syntactical and semantical problems.

The prototype was presented to 21 children (ten boys and eleven girls) with 13 to 14 years of age. The interviews took place in groups of five pupils each (boys and girls separated) and a gender-mixed group of eleven at two secondary schools in Duisburg. The smaller groups were interviewed for about 30 minutes while it took about 70 minutes to cover the agenda with the gender-mixed group. The children had little or no programming experience. We did not interview members of the lower age of our target group for our first feedback round as

we assumed a higher ability to reflect upon the design decisions by the older participants. This fact must be kept in mind when reviewing the following results.

The interviews started with a short introduction and questions about general computer access and computer literacy. We then tried to motivate the purpose of programming by stressing the influence software has on everybody`s private and work life. We tried to constitute programming as a possible appealing and creative activity that enables to independently solve technical based problems. Then the prototype was introduced along with three tasks. The sessions ended with a lively and open discussion as well as a control question about what the pupils learned.

In general, the answers regarding computer access and computer literacy coincided with our research findings as all interviewees used the computer regularly but had only little knowledge about functions or internal components. In comparison, we found that the girls group in general was significantly less interested in the topic compared to the boys or the mixed group. During the practical part, all children were able to quickly understand the meaning of the available commands (e.g. void forward();) as well as the given task to capture the yellow star. Two different ways of solution, visual command window and writing code were introduced. As a result, the children again did very well and got rather excited about their unknown abilities. They were partly able to construct new procedures within the PLE after a short explanation, which resulted in a vivid experimentation session. Surprisingly only little syntax problems occurred, probably because of the simplicity of commands and the supporting visual command window.

During the discussions, children throughout all groups encouraged the close integration of a cover story along with the levels, tasks and also graphic assets but criticized the chosen one as much too childish in nature. The intensity of the subsequent discussion leads to the conclusion that personalization of the environment is quite important to children. This might be connected to the general need for individualism during adolescence. Diverse statements were given regarding the didactical concept, structure, level of difficulty and integration of lecture material and tasks. Some interviewees specifically asked for a strong structure while others stressed the importance of a more open environment, which enables free experimenting and playing. The short interview situation did not allow us to draw any relation between the different statements and the context of the learners. Graphics and UI questions resulted in rather expected statements: Good graphics are important to children. It can be assumed that the more frequent computer users are obviously used to appealing representations from current games and other media. The “unprofessional” graphics of Plumps therefore was rated from “too simple” and “formal” to “trashy”. Furthermore, there was a general agreement upon the need for an as-simple-as-possible interface. The current one did allow the children to easily learn the compile, execute and restart buttons but expectedly failed to clarify the meanings of all Buttons/UI objects visible. Surprisingly, the current text form of story and instructions did not seem to bother children. Some suggested voice playback, though.

When asked how to improve motivation, all interviewees strongly recommended a constantly rising challenge. Furthermore, clarification about relevance outside school was pointed out. Game elements, such as the use of a game metaphor or diverse reward systems (points and leader boards), were considered as very motivating to a majority of pupils, as were possibili-

ties for individualization and user generated content. However, the groups largely agreed that the PLE should not be too much of a game as this would confuse its focal purpose ("we shall learn with it"). As a last remarkable suggestion, the need for competitive and cooperative elements, meaning aspects of social learning, was suggested by the mixed group and resulted in a broad acceptance.

Before closing the interview sessions, group members should try to express what they had learned during the session, which led to quite different results. The girls group was not able to summarize the contents and intention of the session correctly while the boys group did somewhat better. The best feedback was given by the mixed group which might result from different reasons such as size of the group and the twice as long session. Also, the differences likely resulted from the different proficiency levels of the groups, as all in all the mixed group worked more disciplined, more engaged and came up with better overall results, too. Of course, our findings need to be re-assessed by a formal evaluation across all ages of the target group.

4 Conclusions

To sum it all up, we conclude that our first iteration towards the user centered design of a PLE for children was successful. The research results and assumptions regarding the didactical concept and motivational factors seem to be correct and motivate the continuing development towards a playful, tired language PLE for classroom situations. The majority of our focus group acquired basic knowledge about the topic of programming. The well-dosed integration of playfulness and game-based elements in a learning environment seems to work out as we have hoped for but of course needs to be formally validated. But even with the interview sessions being quite informal and far less conclusive than a formal evaluation, we received a lot of transparent and helpful feedback. It was very motivating and encouraging to see pupils building up new knowledge of a complete new topic in very short time. However, it will be challenging to maintain the initial excitement throughout a number of sessions by balancing the integration of motivational game elements and user generated content while not changing the environment itself into a game. This could lead to misconceptions about the nature of programming. Another challenge is the development of a concept that allows an easy transition from the scalable learning environment to a general purpose language, which is one of our main goals.

5 Outlook

Further research in basically all areas is necessary towards the development of the next prototype. In order to structure the next steps we identified three main areas for our future work. First, we will concentrate on the development and integration of a long-term motivational concept with a focus on rewarding game elements, integration of user generated content and

a simplified but supportive GUI with appealing graphics. Along with these improvements, we will develop a concept for a formal evaluation to assess our efforts.

Second, we will focus on an elaborate overall didactical concept. Scope, depth, integration and representation of instructional material will be discussed in close partnerships with computer science teachers as well as other experts.

In a third area, we will work towards an optimal integration of the PLE into class room situations and the curriculum. Currently, we seek to integrate a de-briefing functionality for teachers and learners to reflect a solved task. Also, the idea of a teachers interface for an easy integration of new lessons is being discussed. When considering the classroom being a multi-user environment, more ideas such as pair-programming features or the support of problem-based learning concepts, corresponding with social interaction features, come up.

Acknowledgements

We would like to thank the teachers and pupils of the interview groups of Mercartor Gymnasium and Fermann Gymnasium Duisburg for their great support. Also, we very much thank Dietrich Boles for his friendly support and sharing experiences.

References

Boles, D. (2008). Spielend Programmieren Gelernt mit dem Java-Hamster-Modell. Wiesbaden: B.G. Teubner Verlag.

Charsky D. (2010). Making a Connection: Game Genre, Game Characteristics and Teaching structures. In Van Eck, R. (Ed.): Gaming and Cognition, Theories and Practices from the Learning Sciences, USA: IGI Global. 189-212.

Feierabend, S. Karg, U. Rathgeb, T. (Eds.) (2010). KIM-Studie 2010. Medienpädagogischer Forschungsbund Südwest. http://www.mpfs.de.

Felicia, P., (2009). Teachers Handbook on how to use digital games in schools. European Schoolnet and ISFE.

Grubert, F. (2009). Wie Kinder lernen. Saarbrücken: VDM Verlag.

Horz, H. (2004). Lernen mit Computern. Münster: Waxmann Verlag.

Hundhausen, C.D., Douglas, S.A., Stasko, J.T. (2002). A Meta-Study of Algorithm Visualization Effectiviness. In Journal of Visual Languages and Computing,13. 259-290.

Kelleher, C. Pausch, R. (2005). Lowering the barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers. In ACM Computing Surveys, Vol. 37, No.2. New York. 83-137.

Kerres M. (2001). Multimediale und telemediale Lernumgebungen. Konzeption und Entwicklung. München: Oldenbourg Wissenschaftsverlag.

Klein, B. (2000). Didaktisches Design hypermedialer Lernumgebungen. Die adaptive Lernumgebung „incops“ zur Einführung in die Kognitionspsychologie. Marburg: Tectum Verlag.

Krotz, F., Hasebrink, U. (2003). Medienkompetenz von Kindern und Jugendlichen für die Informationsgesellschaft und ihre Bedingungen in Japan und Deutschland. Hamburg: Arbeitspapiere des Hans-Bredow-Instituts Nr.15.

Landesanstalt für Medien NRW (2010). *Computerspiele und virtuelle Welten als Reflexionsgegenstand von Unterricht*. Düsseldorf: LFM.

Papert, S. Mindstorms (1984). *Children, Computers and Powerful Ideas*, *2nd Edition*. USA: Basic Books.

Pea, R.D. Kurland, D. (1984). *On the cognitive effects of Learning Computer Programming*. In: *New Ideas in Psychology Vol.2*. Amsterdam: Elsevier Ltd. 137-168.

Powers et al. (2006). *Tools for teaching introductory programming: What works?* In: *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*. USA: ACM. 560-561.

Reichert, R. Hartmann, W. Nievergelt (2001). *Kara, finite state machines and the case for programming as part of general education*. In: *Proceedings of the 2001 IEEE Symposia on Human-Centric Computing Languages and Environments*. Italy: Stresa. 135-141.

Resnick et al. (2009). *Scratch: Programming for All*, In: *Communications of the ACM*. Vol. 52 No.11, 2009, 60-67.

Schwill, A (2001). *Ab wann kann man mit Kindern Informatik machen? Eine Studie über informatische Fähigkeiten von Kindern*. In: Keil-Slawik, R. Magenheim, J. (Eds.): *INFO ´01 Informatikunterricht und Medienbildung*. Gesellschaft für Informatik. 13-30.

Sheehan, R.J. (2005). *The Icicle Programming Environment for Children*. Doctoral Thesis. USA: University of Auckland

TNS Infratest (2011). *Bildungsstudie: Digitale Medien, Initiative D21*. Sonderstudie im Rahmen des (N)Onliner Atlas 2011. Cornelsen Verlag GmbH, Texas Instruments GmbH, Initiative D21 e.V.

Wölke, S. (2009). *Spielen und Lernen – das Lernspiel als Widerspruch in sich?!* München: Grin Verlag.

Contact Information

Dipl.-Medieninf. Ralf Schmidt
Universität Duisburg-Essen
Entertainment Computing Group
Fakultät Ingenieurwissenschaften
Forsthausweg 2
D-47057 Duisburg

Tel.: +49 (0)203 379-2440
E-Mail: ralf.schmidt@uni-due.de
WWW http://medieninformatik.uni-due.de