Playful generation of game mechanics

Jörg Niesenhaus

In-play production of rules in an experimental game prototyping framework enables users to implement and test their game ideas within minutes.

End-user development and user participation in the development of digital games has existed for as long as games themselves, but the degree of participation has significantly increased in recent years. In the past, modifying a game was only possible for skilled programmers using tools such as hex editors that allow manipulation of the binary game files. Nowadays, several developers and publishers offer games with a focus on user-generated content. Although significant effort has gone into developing tool sets for creating this type of content, most games only offer user participation in graphical development and the design of levels.

In our work, we have implemented a prototype development framework that allows end-users to define game rules in a playful manner. By ‘playful’ we mean that new rules can be defined while playing the game, thus extending the range of behaviors in the game world dynamically while the program is run. Furthermore, we provide a sensor-action framework that allows users to create a wide range of self-defined events and to specify actions that handle these events. We have designed all interactions necessary to define rules in a strongly visual, direct-manipulation style, applying principles of visual programming and programming by demonstration.

Burnett defines visual programming as a programming style in which more than one dimension is used to convey semantics. The term is related to programming by demonstration, which describes a system being able to infer the program structure based on the user’s inputs, recognizing patterns, and applying them to an algorithm. There are several popular visual programming environments in research and on the commercial market, including Alice, AgentSheets, Scratch, StarLogo TNG (The Next Generation), and Kodu.

Our game prototyping framework, called 2DGree, features interchangeable components to provide a flexible and controlled environment for evaluating different interaction techniques, as well as for testing different methods of visual programming and programming by demonstration. To lower the entry barrier for the user, the framework focuses on the in-play definition of rules rather than on creating elaborated, visually appealing graphics. For this reason, we limit the framework to a 2D game world.

After setting up the game world and putting game entities (e.g., characters or vehicles) in place, the user can attach a range of software sensors to static or dynamic objects, which trigger events or track data: see Figure 1. A software sensor serves as a visual and formal representation of an event generator, which is able to track different types of user or non-player character interactions in the game world. Such an interaction may happen if the player collides with a static object or enters a certain geometric area around a specific entity. Our framework represents these sensors through basic geometries, which are either generic or represent some specific type of ‘senses’ such as vision or hearing sensors.

Figure 1. The 2DGree sensor editor with a 2D landscape and basic sensor entities.

Continued on next page
To generate events, users can instantly start the play mode, which puts them in control of the player entity. This mode allows the users to explore the different possible interactions by navigating through the game world. In addition, the user can immediately trigger specific events, for example, by dragging an object on top of another one or into an active sensor area. When an event is detected, a context window opens in which the user can select the action to be triggered for this event, such as manipulating entity properties (e.g., health points of characters), deleting or spawning new entities, or linking the event to more complex game mechanics.

In summary, our game prototype development framework, 2DGree, aims to evaluate different methods of visual programming and interaction techniques with regard to their suitability for end-user development in the area of game development. Some of the core interaction techniques and features of this framework include software sensors and the in-play rule generation through methods of programming by demonstration. These features not only save time during the development of game prototypes but also offer users a more intuitive introduction to the generation of game rules. Future work will include further lab studies using a larger variety of components and a large-scale online test to evaluate the implemented methods of direct manipulation and programming by demonstration with a larger number of subjects, including both professional game designers as well as community users. In addition, we will also compare existing prototyping tools and further visual programming environments with the tools incorporated in the 2DGree framework.

Author Information

Jörg Niesenhaus
Centigrade GmbH
Mülheim an der Ruhr, Germany

Jörg Niesenhaus coordinates activities at the North-West branch of Centigrade, a service provider for user interfaces in Germany. Previously, he worked more than 11 years for game companies like Blue Byte and Ubisoft and at the human-computer interaction research group at the University of Duisburg-Essen.

References