Design and Evaluation of Parametrizable Multi-genre Game Mechanics

Daniel Apken, Hendrik Landwehr, Marc Herrlich, Markus Krause, Dennis Paul, Rainer Malaka

To cite this version:

Daniel Apken, Hendrik Landwehr, Marc Herrlich, Markus Krause, Dennis Paul, et al.. Design and Evaluation of Parametrizable Multi-genre Game Mechanics. 11th International Confernece on Entertainment Computing (ICEC), Sep 2012, Bremen, Germany. pp.45-52, 10.1007/978-3-642-33542-6_4 . hal-01556140

HAL Id: hal-01556140

https://inria.hal.science/hal-01556140

Submitted on 4 Jul 2017

Design and Evaluation of Parametrizable Multi-Genre Game Mechanics

Daniel Apken[1], Hendrik Landwehr[1], Marc Herrlich[1], Markus Krause[1], Dennis Paul[2], and Rainer Malaka[1]

[1] Research Group Digital Media, TZI, University of Bremen
[2] Interaction and Space, University of the Arts Bremen

Abstract. Designing digital games is primarily interaction design. This interaction manifests as a meaningful change in the game world. An aspect of a game can only change dynamically with a parametric model of this aspect available. One aspect of digital games is yet missing such a systematic description: the genre of a game is currently only determined by its designer. This paper introduces a new approach that allows for dynamic blending between genres. We describe a set of game mechanics that express the characteristics of different game genres. We extract a parametric model from these mechanics to allow dynamic blending. The paper illustrates the possibilities of this approach with an implementation of a multi-genre-game. It also provides empiric evidence that the described model successfully generates different game genres.

Keywords: multi-genre games, genre blending, parametrizable game mechanics, game mashups

1 Introduction

The design of many computer games is based on common and very stereotypical genre patterns. The mechanics of different first-person shooters, role-playing games (RPG), or platform games are very similar in comparison. Exceptions are so-called "genre mash-ups", e.g., games combining puzzling with action or strategy with RPG. Not only does combining well-known game patterns in new ways give existing genres an interesting twist but this could also broaden the target audience of a game by providing a more personalized game experience. Imagine the player being able to tailor the specific mix of game mechanics to her taste and mood or the game automatically adapting to the player's preferred playing style over time.

However, currently it is exclusively the designer that decides on the specific mix, which is fixed after shipping the game. At the same time, it is important to not take all creative control away from the designer because this would clearly diminish the game experience as good game design is anything but random and arbitrary. The goal is to strike a balance between creative control and influence of the designer and customizability and flexibility regarding mechanics and user preferences. What is needed are fundamental building blocks that allow designers to incorporate dynamic genre mixing and switching in their games in a systematic, controlled, and understandable way. Such building blocks should be easily customizable, ideally using only a floating point variable to switch or blend between different game characteristics. They should facilitate clearly distinguishable play styles at an equal level of quality for each style. From an engineering perspective they should generalize to a wide variety of games.

In this paper we introduce our approach to customizable multi-genre game mechanics. We present the design and evaluation of three specific multi-genre game mechanics: movement metaphor, puzzle factor and AI factor. We show how our multi-genre game mechanics fulfill the requirements discussed above. We report on a conducted user study, providing empirical evidence that our mechanics successfully support different play styles while maintaining a near constant and high level of quality of the overall game experience. We discuss how our approach is generalizable to other games and mechanics.

2 Related Work

Numerous computer games already use procedural game elements. Automated map and asset creation are popular techniques. A very successful example of procedural world creation is Minecraft [3]. It uses an algorithm that imitates nature by heavily relying on perlin-noise [6] to create structures such as fields, hills, or lakes. Other games, e.g., the Diablo series[3], create procedural worlds from precreated interchangeable pieces to convey unpredictability. "Rogue-like" games in general often use similar techniques to create seemingly unique items by randomly selecting from a predetermined attribute range, e.g., a sword with a special attack type, damage value, or player class requirements [12]. Other systems rely on behavioral models such as "rhythm groups" [9, 10]. Search-based content generation [11] or difficulty calculating algorithms [2] are examples of yet another class of related methods. The appearance of assets can also be altered using the mentioned methods, applying slight differences to otherwise similar objects [13]. Façade is an interactive drama [5] employing different techniques, e.g., natural language understanding, to alternate the progress of the story depending on the player's choices. Lopes et al. [4] describe adaptive games, in which parts of the game are altered according to the user's behavior. In some games, e.g., Mario Kart[4] or Max Payne[5], the difficulty is altered based on the player's performance, e.g., providing increased aiming assistance if the player dies too often.

The mentioned works cover procedural content for game worlds, levels, assets, their graphics, and story. However, to our knowledge, there are no examples of games that allow to vary fundamental game mechanics such as the movement metaphor based on a continuous parametrization.

[3] http://www.blizzard.com/games/
[4] http://www.mariokart.com/wii/launch/
[5] http://www.rockstargames.com/maxpayne/

3 Multi-Genre Game Mechanics

A prototype has been developed that provides customizable fundamental game mechanics based on easily changeable numerical parameters. The main goal is to provide game mechanics that can be altered at any time, resulting in clearly distinct game types or genres while at the same time maintaining a high quality and comparable overall game experience. In effect that means that it should be possible to generate games of different genres or blends of such games at the same quality level, which would still work as self-contained games, including all the usual elements of games [7, 8].

3.1 Prototype Genres

While our mechanics can in principle be varied "continuously" as they are represented by floating point parameters, for the prototype and the following user tests we developed three distinct parameter sets, each corresponding to a specific setup as listed below. We selected three popular and distinct genres we wanted to emulate by customizing our mechanics. The goal was to potentially appeal to a variety of player types [8].

Platformer setup. The primary aim of this game variant is to jump and collect items. The player is automatically moved to the bottom. With increasing height the items being collected are also increasing in value.

Shoot 'em up setup. The primary aim of this game variant is to shoot enemies while dodging obstacles. The player is automatically moved to the right. Enemies are getting stronger as the player progresses.

Puzzle setup. The primary aim of this game variant is to solve puzzles by changing the color of at least three hexagons. The player is not automatically moved. Puzzle difficulty increases with each one solved.

3.2 Static Mechanics

In addition to the customizable "dynamic" mechanics described below, all games (or game variants) include basic "static" mechanics that work similarly independent of any parametrization. This includes a collecting mechanic and a score system. Collecting items, killing enemies, and solving puzzles yields the player points, which provide a measurable outcome and goal to the player.

Additionally, we devised a simple interaction mechanic independent of the game variant. The player can click on game objects to make the game avatar launch box-shaped projectiles at these objects. This "shooting" mechanic is static across variants, however, the result is based on the respective target object type, which may vary. In effect this allows the player to remove obstacles, attack enemies, or solve puzzles using the same basic interaction method.

3.3 Dynamic Mechanics

The principle of dynamic mechanics, being controlled by parameters, has been applied to several core mechanics of the prototype. One of the most fundamental mechanics is movement. In principle, movement can be customized through the following parameters: direction, acceleration, speed, automatic or manual movement, linear or non-linear movement. Furthermore, movement possibilities are heavily dependent on the surrounding game world and objects, e.g., types of barriers, walls, or platforms. The specific type of movement and obstacles have a great impact on the perceived game genre and are not independent. In order to parametrize movement for different genres we investigated common movement patterns [1]. We developed a special kind of barrier which is only passable from one side and which is automatically rotated according to the movement parametrization. Although other movement parameters (listed above) are also slightly adjusted according to the specific setup, the movement metaphor is mainly parametrized by a single floating point parameter – "gravity" – that determines the direction of forced or non-forced (by setting gravity to zero) player movement. The barriers as well as other movement parameters and player controls are adapted according to this parameter.

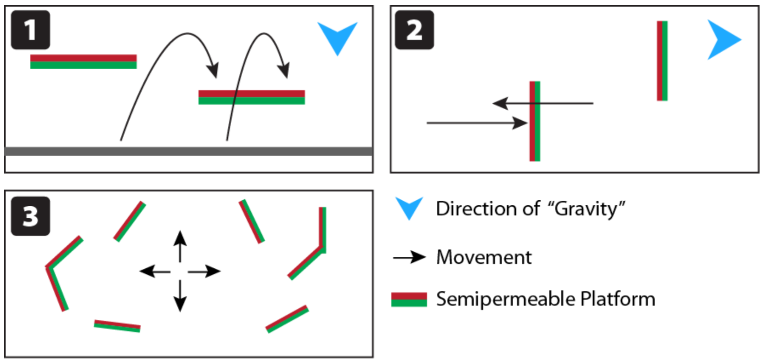
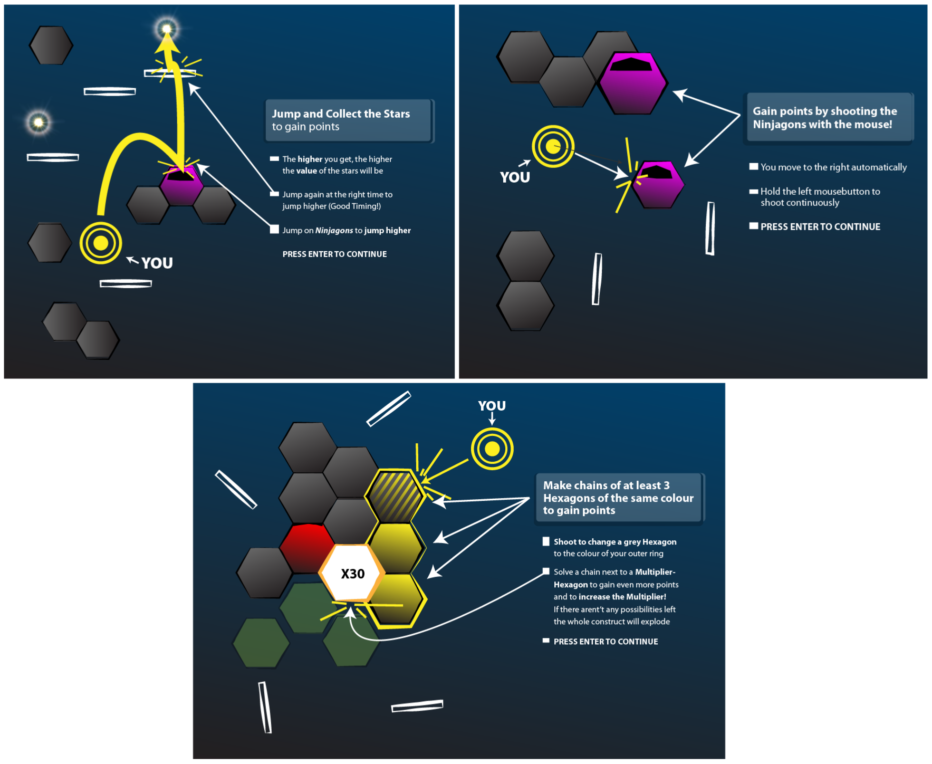


Fig. 1. Three different movement and obstacle combinations controlled by "gravity" are used to model three different game genres.

In the platformer setup (figure 1, box 1), the gravity is directed downwards. As the desired direction in this setup is upwards, the player can use a jumping motion to get there. The platforms align according to the gravity, allowing the avatar to stand on top of them and jump through them from below. Additionally, movement to the left and right is not impeded. The gravity in the shoot 'em up setup (figure 1, box 2) draws the player rightwards. The goal is to shoot and avoid enemies, which is the easiest when moving with the gravity. The avatar can move up and down freely. The platforms serve as obstacles that impede the players progress, instead of supporting it. In the puzzle setup (figure 1, box 3), there is no gravity at all, which means the avatar can move freely in all directions.

In this case, it cannot be predicted where the player will navigate next, so the platforms "align" in random directions.



Jump and Collect the Stars to gain points

- The **higher** you get, the higher the **value** of the stars will be
- Jump again at the right time to jump higher (Good Timing!)
- Jump on ***Ninjagons*** to **jump higher**

PRESS ENTER TO CONTINUE

Gain points by shooting the Ninjagons with the mouse!

- You move to the right automatically
- Hold the left mousebutton to shoot continuously
- **PRESS ENTER TO CONTINUE**

Make chains of at least 3 Hexagons of the same colour to gain points

- **Shoot to change a grey Hexagon** to the colour of your outer ring
- Solve a chain next to a **Multiplier-Hexagon** to gain even more points and to **increase the Multiplier!** If there arent't any possibilities left the whole construct will explode
- **PRESS ENTER TO CONTINUE**

Fig. 2. Instruction screenshots for the three different game variants modeled in the prototype: Platformer (top left), Shoot 'em up (top right) and Puzzle (bottom).

Another fundamental mechanic is the type of non-player objects/agents the player can interact with. To customize objects to a wide variety of uses they should be of a simple shape, while still providing enough possibilities to give them an interesting appearance and functionality. Furthermore, they should work as individual objects but they should also be combinable into larger clusters. We found that simple hexagons fulfill these requirements. They can represent obstacles, enemies, or puzzle pieces (hexagons of different colors), depending on the intention of the player and/or goal of the game (figure 2). This is realized by providing a certain behavior when creating new hexagons in a level. Most hexagons being spawned are neutral and act as obstacles or platforms to stand or jump on. Two parameters called "puzzle" and "aggressiveness" factor, respectively, are used to alter this spawn rate, providing more objects of either kind. Additionally, the aggressiveness is used to determine the initial behavior of enemies, making them more passive or active and the puzzle factor determines the puzzle complexity as it influences the number of colors used.

Enemies that are being hit become more aggressive, approaching the player faster and stealing points when reaching her. All other hexagons that are hit (except for special puzzle hexagons) adopt the color of the projectile. When three or more hexagons of the same color are connected to each other, they explode and the player gains points but only a few. When a special puzzle hexagon (a hexagon showing a number) is connected to them, the points gained are multiplied by its number and the number itself increases, making it possible to gain even more points when destroying other hexagons connected to it. Enemies that are not aggressive at all will not approach the player and cost her no points when she touches them. However, they still push her off, making them ideal to be used as trampolines, supporting the jumping mechanic. Larger formations of neutral hexagons can often impede the player's way. The puzzle and shooting mechanics can be used to connect three of them, make them explode and clear the way.

4 User Study

An online user study was conducted to empirically confirm the suitability of the described approach. The main goals of the study were to confirm that the multi-genre game elements actually work for each genre and that at the same time each genre is still clearly distinguishable by the user.

The game was implemented in Java and could be downloaded from our website. We invited users to participate by circulating the access link both over internal and external mailing lists and social network sites of our university. In the disclaimer, users were instructed that we wanted to test new game designs but no specifics on multi-genre or generative game design were provided. We used a within-subjects design with each user playing all three game variants (platform, shoot 'em up and puzzle setups as described in section 3.1). The order of game variants was randomized using a latin square across participants. Participants were asked to select the three most important concepts specific to each game variant from a list of 14 items (see below). They were asked to rank all variants in comparison and additionally to rate each individually on a 5-point Likert scale. We collected similar ratings on graphical quality, story, and game play for each version.

Overall 64 people participated in the study. After removing invalid data, we still retained 44 valid data sets for further analysis. Of these final 44 participants, 37 were male and 7 female with an average age of 25.66 (SD 4.2) years. Friedman tests revealed no significant differences between the three game variants regarding ranking, ratings (overall, graphical quality, story, gameplay), perceived fun, learnability of the controls, and clarity of the game goal. Mean values for perceived fun (5-point Likert scale; 1 = better; 5 = worse) were 2.52 (SD 1.0), 2.45 (SD 1.13), and 2.32 (SD 0.983). Overall rating mean values were 2.61 (SD 0.754), 2.55 (SD 0.761), and 2.45 (SD 0.820). Mean values for learnability of controls (5-point Likert scale; 1 = worse; 5 = better) were 4.75 (SD 0.534), 4.73

(SD 0.694), and 4.70 (SD 0.701). Mean values for clarity of gaming goal (5-point Likert; 1 = clear; 5 = unclear) were 1.52 (SD 0.628), 1.45 (SD 0.875), and 1.77 (SD 0.886). Application of Cochran's Q revealed significant or highly significant differences between the three game variants for the following concepts: logical thinking ($\chi^2(2) = 33.583$, $p < 0.01$), reaction speed ($\chi^2(2) = 11.760$, $p < 0.01$), collecting ($\chi^2(2) = 29.727$, $p < 0.01$), shooting ($\chi^2(2) = 26.963$, $p < 0.01$), jumping ($\chi^2(2) = 70.205$, $p < 0.01$), problem solving ($\chi^2(2) = 17.077$, $p < 0.01$), aiming ($\chi^2(2) = 14.519$, $p < 0.01$), dodging ($\chi^2(2) = 8.667$, $p < 0.05$), and attacking ($\chi^2(2) = 24.4$, $p < 0.01$). For maneuvering, tactical thinking, hand-eye coordination, concentration, and exploration the differences were found to be not significant, although the p-value for tactical thinking ($p = 0.07$) missed the 5% threshold very narrowly.

5 Discussion

As reported above, no significant differences have been found between the three game variants in relation to perceived quality and fun. All variants were comparable in terms of ranking, ratings and sub-ratings (such as graphical quality etc.). The absolute ratings reinforce the results of the relative rankings and they additionally show that all variants were not just rated equally, but equally good. This also pertains to the learnability of the controls. On the other hand nine of 14 concepts exhibited statistically significant differences. This shows that the participants were indeed very aware of differences in game play and that they very clearly attributed different concepts and core mechanics to each game instance.

In summary this shows that our approach successfully led to the creation of three very distinct and different games while maintaining a good quality and working game mechanics for each game individually. Switching between these games – or genres – is now just a matter of adjusting the three main parameters (gravity, puzzle, and aggressiveness), which are floating point variables and thereby also enable simple blending between different variants.

Our participant base was biased towards young, male, hard-core gamers. The results of the study are therefore limited to this audience at the moment. However, one could also interpret this audience as being an audience of computer game experts.

6 Conclusion and Future Work

With regard to our initial research question, the results of the reported user study show that we were able to successfully create three distinct gaming experiences the participants clearly related to three different game genres while maintaining a comparable level of high quality across all game variants.

Our approach currently rests on three main pillars: a variable motion mechanic, the puzzle factor, and the AI (or aggressiveness) factor. All games that feature the exploration of space could potentially benefit from dynamic motion mechanics. What is needed to implement genre-blending is a "natural" mapping of a floating point game attribute to a motion metaphor, such as gravity in our case. Examples could be other physical properties related to motion like friction, buoyancy, or permeability. Our puzzle factor can be generalized to dependencies between other game elements. Thus increased puzzle factor means more dependencies. Again a mapping function is needed that maps a floating point variable to more complex dependencies. The AI factor influences if the player perceives other objects as static object, neutral agents, or enemies. This maps rather directly to some sort of "aggressiveness" factor in most simple game AI systems.

So far we only tested three distinct game variants. In the future we plan to add additional variants to investigate further the scalability of our approach. The parametric approach in principle allows to blend game variants at runtime. When and how to blend has to be determined and evaluated to ensure a convincing gaming experience.

References

1. Bjork, S.: Patterns in Game Design. Charles River Media, Hingham Mass., 1st ed. edn. (2005)
2. Compton, K., Mateas, M.: Procedural level design for platform games. In: Proc. AIIDE'06. pp. 109–111 (2006)
3. Handy, A.: Markus 'Notch' Persson talks making Minecraft. http://www.gamasutra.com/view/news/27719/Interview_Markus_Notch_Persson_Talks_Making_Minecraft.php [Last accessed: 20.04.2012] (2010)
4. Lopes, R., Bidarra, R.: Adaptivity challenges in games and simulations: A survey. IEEE Transactions on Computational Intelligence and AI in Games 3(2), 85 –99 (June 2011)
5. Mateas, M., Stern, A.: Facade: An experiment in building a fully-realized interactive drama. In: Proc. GDC'03. San Jose, CA (2003)
6. Perlin, K.: Improving noise. In: Proc. SIGGRAPH'02. pp. 681–682. ACM, New York, NY, USA (2002)
7. Salen, K., Zimmerman, E.: Rules of Play: Game Design Fundamentals. MIT Press, Cambridge Mass. (2003)
8. Schell, J.: The Art of Game Design: A Book of Lenses. Elsevier/Morgan Kaufmann, Amsterdam; Boston (2008)
9. Smith, G., Cha, M., Whitehead, J.: A framework for analysis of 2d platformer levels. In: Proc. SIGGRAPH'08 Symposium on Video Games. pp. 75–80. ACM, New York, NY, USA (2008)
10. Sorenson, N., Pasquier, P., DiPaola, S.: A generic approach to challenge modeling for the procedural creation of video game levels. IEEE Transactions on Computational Intelligence and AI in Games 3(3), 229 –244 (Sept 2011)
11. Togelius, J., Yannakakis, G., Stanley, K., Browne, C.: Search-based procedural content generation: A taxonomy and survey. IEEE Transactions on Computational Intelligence and AI in Games 3(3), 172 –186 (Sept 2011)
12. Toy, M., Wichman, G., Arnold, K., Lane, J.: Rogue by artificial intelligence design (1983)
13. Watson, B., Muller, P., Wonka, P., Sexton, C., Veryovka, O., Fuller, A.: Procedural urban modeling in practice. IEEE Computer Graphics and Applications 28(3), 18 –26 (May-June 2008)