The AIIDE-17 Workshop on
Experimental AI in Games
WS-17-19

ProcDefense — A Game Framework for Procedural Player Skill Training

Brandon R. Thorne, Hiru Nelakkutti, Joseph Reinhart, Arnav Jhala
Department of Computer Science
North Carolina State University
Raleigh, NC
{brthorne | hsnelakk | jareinha | ahjhala}@ncsu.edu

Abstract

A challenge of game design is in providing affordances to players so that they can learn and improve their skills. Advances in Procedural Content Generation (PCG) suggest this type of game content is a candidate for automatic creation. Some work in PCG has been successful in customizing game difficulty to achieve desired player experience; however, this often involves bringing the difficulty of the game to a level appropriate for the player's current skills. Players desiring to improve their performance in a particular game may be willing to tolerate relatively higher levels of frustration and anxiety than are targeted in experience-based approaches. As an initial step in this line of inquiry, we introduce ProcDefense, an action game with a modular difficulty control interface, as a platform for future inquiry into the effectiveness of differing PCG techniques for performance-training, dynamic difficulty adjustment.

In the video games industry, designers are challenged to support player mastery of gameplay mechanics through the game interface. They employ a variety of techniques to introduce the mechanics including tutorial levels, diegetic commentary, and extradiegetic hints. Assisting players in attaining mastery once the mechanics are understood is often done with solo practice modes or scoring feedback in addition to the sequence of challenges comprising the game itself. With the advances in real-time player modeling and procedural content generation, it has become possible to consider approaches that could dynamically tune gameplay parameters with the objective of improving player skill.

Procedural Content Generation (PCG) research is often concerned with customizing those challenges in order to elicit particular affective responses from a player (Yannakakis and Togelius 2011). This work has made excellent progress in describing system behaviors which can increase player enjoyment and engagement with a game; however, this is often accomplished by adjusting challenges to match player skill. In more competitively oriented games, adjustment of content only with respect to player experience may not aid the player in achieving their goal. In this case player enjoyment and engagement are still important, but aiding the player in improving their skill in the game must also be considered. In other applications, educators are using simulation-based training for tasks requiring user acquisition of domain knowledge and manipulation of interface elements. This can be seen in flight simulators and, more recently, augmented reality surgical training (Evans and Schenarts 2016). Their success suggests a similar approach of simulating game situations to train players might prove effective. We theorize that by incorporating models of player experience and engagement with simulationist strategies for identifying and presenting training scenarios to the player, we can guide a PCG system to present those training scenarios to the player within the original game environment, accelerating improvement of the player's performance over undirected game practice alone.

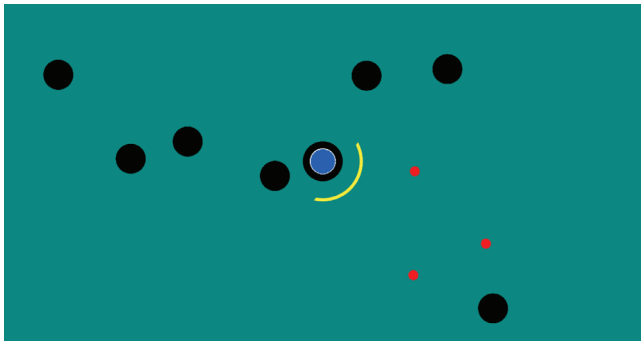


Figure 1: ProcDefense screen capture depicting game state after shattering a projectile. Smaller, short-lived projectiles replace the one destroyed.

In this paper we describe the mechanics of a simple game, ProcDefense, to be used as a platform for testing the relative efficacy of different PCG approaches for such player training. ProcDefense draws on current work in PCG to inform construction of a platform for explicitly tackling the challenge of tweaking game parameters in support skill development across multiple dimensions of skill.

ProcDefense

ProcDefense is an action game in which the player uses an arc-shaped paddle to defend a central core from incoming projectiles. The player can move the paddle along a circle at a fixed radius from the center of the core. Projectiles are con-

tinuously spawned which are initially directed toward the core. The projectiles begin with a set durability that is decremented every time the projectile is reflected from the paddle, the bounds of the play area, or other projectiles. When the durability of a projectile reaches zero, it is destroyed and player score is increased. If a projectile reaches the core, the projectile is destroyed and the player score is reduced. After several successful reflections, the player may change the mode of the paddle to shatter the next incoming projectile as shown in Figure 2. This destroys it immediately and spawns several projectiles with low durability which destroy any other projectiles they meet. A level in this game is defined as a set of difficulty parameters (Table 1) and a set of completion criteria. Difficulty parameters include the set of projectile types as well as paddle properties. A given projectile type is spawned as an instance of its associated type parameters shown under projectile type in Table 1. The completion criteria dictate the rate of score increase required and the duration over which it must be maintained in order for the player to complete the level.

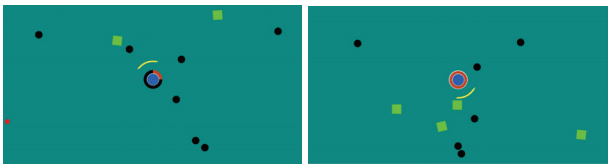
The game provides an interface through which the current difficulty parameters may be adjusted. Which parameters to change and how adjustments should be made in order to facilitate player skill improvement is the core topic of inquiry this platform supports. For example, a training subsystem might reduce projectile speed if players are losing score quickly, but in order for the player to proceed to the next game level, the difficulty settings must be at least as difficult as the level specifies. The player must then maintain the rate of score increase for the duration specified in the completion criteria in order to move to the next level. If the player is never able to satisfy the completion criteria at the target difficulty, they will remain within their current level whose parameters will continue to be adjusted to assist them in improving.

Platform and design considerations

While there exist several games with AI interfaces, particularly related to AI competition frameworks, we designed a new game for inquiry into difficulty-adjusting, training systems. We needed control over both the design and the source of the game so that design decisions could be made with the dynamic training aspect in mind. This consideration indicated that at least an open-source game was preferable. We also needed freedom in determining what features of game state are available for consideration and how often they are reported, as well the ability to evolve game mechanics as we explore the interaction of difficulty adjustments and skill improvement. Given this freedom, we could ensure that our game had sufficient variety in play skill as well as use of high-level strategies.

Among open-source games, we considered a simplified real-time strategy game *MicroRTS* (Ontanon 2013), but due to its domain complexity, detecting particular behaviors in need of remediation is challenging. There are a large number of strategies which might be successful in a given situation, and we cannot necessarily tell which the player is attempting to employ. We also considered the Mario AI competition platformer implementation (Togelius et al. 2013) as a candidate game; it has a simpler domain, and there has been a wealth of level generation research conducted with this game in recent years (Pedersen, Togelius, and Yannakakis 2009; Togelius et al. 2011; Shaker et al. 2011); however, we were concerned the popularity of the platformer genre might have some confounding effect on future experimental results. Likely many of our volunteer participants for a user study will have experience with standard platformer controls. If we use the standard controls as the player interface, our players are likely to have little to learn, and if we deviate from standard controls in order to train the player in their use from novice to master, we violate the genre conventions and player expectations leading to frustrated participants.

In light of these considerations, we opted instead to design a new game with a simple domain that is perhaps less likely to already have extensive time investment from our participant pool. We incorporate the shattering mechanic to add a strategic element to player decision making. Instead of always greedily moving the paddle to block the most imminent threat, the player may also choose timing and positioning to eliminate sets of projectiles. This gives some variety in strategic decision making within a simple and intuitive framework.



(a) Charging (b) Fully Charged

(c) Shattering Activated (d) Shatter Complete

Figure 2: Ability charge progression for shattering paddle. Paddle deflections build charge which may be used to destroy one projectile.

Interface

We provide to the difficulty adjusting subsystem a game state description each time durability of a projectile is reduced. This includes type, position, velocity, and durability of each projectile; paddle position, size, and speed; and current level completion progress. The attached subsystem is then free to use or ignore any of the provided features, affording us opportunities to compare feature selection strategies in future studies. The subsystem is able to manipulate the current difficulty settings of the level (see Table 1), while the player's performance required to complete the level remains fixed.

Category	Parameter	Description
Environment	Spawn Interval	Minimum duration between spawn of any projectiles.
	Projectile Types	Set of projectile types which can be spawned.
Projectile Type	Scale	Size of projectile sprite.
	Spawn Interval	Duration between spawn of projectiles of this type.
	Initial Speed	Speed upon projectile spawn.
	Durability	Number of collisions required to destroy this projectile.
	Damage	Score reduction upon this projectile type hitting the core.
Paddle	Size	How much of the core the paddle protects.
	Speed	How fast the paddle rotates about the core.

Table 1: Difficulty Parameters

Future Work

With the game in place, we can begin development on the systems to drive difficulty adjustment. One set of possible system components is given as an example.

The system could be driven by a subset of the features of the game state which are indicative of poor performance. Feature selection might be learned from gameplay captures, crowdsourced from players, or performed by individual domain experts. Correlating features to mistakes in gameplay would allow for targeted application of strategies for adjusting difficulty to remediate the mistakes. The specific strategy chosen might depend upon a model of the individual player's susceptibility to frustration and their motivation to improve.

On the game platform itself, adding functionality to include or exclude game mechanics in a given play session would allow us to examine the effect of the player's cognitive load on training efficacy. Yielding some insight into whether the overall time required to master a level could be improved by allowing the player to practice a skill in isolation, or if reintegrating that skill into a more complex game would take just as much time as training it in a complex setting in the first place.

ProcDefense is a platform that supports inquiry into procedural player skill training. It is a tool to test strategies for dynamically adjusting game difficulty in order to accelerate player performance improvement.

References

Evans, C. H., and Schenarts, K. D. 2016. Evolving educational techniques in surgical training. *Surgical Clinics of North America* 96(1):71–88.

Ontanon, S. 2013. The combinatorial multi-armed bandit problem and its application to real-time strategy games. In *Ninth Artificial Intelligence and Interactive Digital . . .* , 58–64.

Pedersen, C.; Togelius, J.; and Yannakakis, G. N. 2009. Modeling player experience in super mario bros. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, 132–139. IEEE.

Shaker, N.; Asteriadis, S.; Yannakakis, G. N.; and Karpouzis, K. 2011. A game-based corpus for analysing the interplay between game context and player experience. In *Affective Computing and Intelligent Interaction*. Springer. 547–556.

Togelius, J.; Kastbjerg, E.; Schedl, D.; and Yannakakis, G. N. 2011. What is procedural content generation?: Mario on the borderline. In *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*, 3. ACM.

Togelius, J.; Shaker, N.; Karakovskiy, S.; and Yannakakis, G. N. 2013. The mario ai championship 2009-2012. *AI Magazine* 34(3):89–92.

Yannakakis, G. N., and Togelius, J. 2011. Experience-Driven Procedural Content Generation. *IEEE Transactions on Affective Computing* Vol. 2(No. 3):147–161.