Monopoly

Postmortem

Daniel McFerren

POSTMORTEM

Over the course of the last semester, we embarked on an ambitious attempt to implement Monopoly in the Vicious Engine – specifically in a format appropriate for Sony's PlayStation Portable. Even given that it was a classroom environment, a lot was learned.

SUCCESS 1: CODE ADAPTABILITY

I was fairly successful at adapting to Vicious Engine's style of coding. I have some experience in shoehorning modern programming concepts into awkward and legacy systems, so some of this didn't produce as much of a challenge as it might have otherwise. I can especially appreciate a token-based coding system when it comes to compiler efficiency and refactoring (renaming variables, assets, or functions after they've been implemented). The code produced was, for the greater part, highly functional.

SUCCESS 2: MOST RULES COVERED

I was able to implement a majority of the official Monopoly rules. The original plan had game features divided by how integral they were to the core Monopoly game, but the actual development process focused on a sequence more related to the difficulty and features of the engine rather than the game. Despite this, I was able to implement pretty much everything that can happen to a player when they play a game of Monopoly, if you ignore the context of the other players and inter-player interactions.

SUCCESS 3: UNIQUE ASSETS

I was able to generate unique assets – both 2D and 3D – and have them operate fully in the Vicious Engine. The assets were a bit clumsy and closer to placeholder art than final game assets, but they were effective and didn't take long to implement. Had I taken more time, I think it's possible I would have managed to produce much more polished and textured assets, for a better player experience.

IMPROVEMENT 1: ENGINE KNOWLEDGE

I wrote the Game Design Document without near enough knowledge of the game engine. Even a cursory knowledge would probably have radically changed the way I looked at the project, the UI style and features I had planned, and the process I used for development. I learned that the choice of an engine goes far beyond the most commonly-advertised points – high-powered graphics – and is instead largely a question of administrative impact. It doesn't matter if the engine can render five million polygons per frame if the artists can't get meshes into the engine and programmers can't get the renderer to work before the deadline.

IMPROVEMENT 2: PROPER PLANNING

The scope of the project didn't consider limits on schedule or resources. In addition, I didn't adapt the plan as reality confounded my ambitions. Even on a one-person project, maintaining design documentation, or taking note of the things I expected to be done so I could keep the critical items on my list, would have produced a much more focused result. In addition, I didn't take advantage of knowledge of the publisher's milestone requirements (the checklists whereby we were to be graded) in my project planning – though I had most items done on time, many of them were an afterthought. Had I been working with a publisher rather than an instructor, and for money rather than grades, it would have literally paid off to work to their expectations rather than my own estimations.

IMPROVEMENT 3: KNOWLEDGE OF AUDIENCE

I made assumptions about the game's audience without doing any research. While there wasn't a defined target demographic, I made the assumption that the game would be of 'killer app' status, or be part of the PSP's original packaging – that people would adopt the system to play this version of Monopoly. In addition, I did next to no research on interface standards and player expectations when designing or implementing the interface. Such a lack of knowledge of the future customer base may have slipped past a publisher or even a marketing department, but it certainly wouldn't have gotten past the player base.

IMPROVEMENT 4: UI STYLE

I am very unpracticed in putting together stylish user interfaces. Even given that, the interface I presented was extra clunky and awkward. For anything but experimental prototypes or programmer-focused demonstrations, it is critical to build and maintain a clean, focused interface. This is especially important in a live development environment, when any given 'operational' prototype can become the next trade show demo or early industry magazine review. Overall, UI and UX took a back seat to rules and technical implementation – not a strategy for long-term success in game development.

IMPROVEMENT 5: SHODDY TESTING

I didn't go very deep with any kind of testing (as evidenced by my demonstration). I had no testing plans, or test flow diagrams, or record of what had been built and how it might interact with anything else after it was done – as a result, I couldn't even predict where bugs would appear, let alone test for them with any accuracy. At the least, I should have implemented some standard '0, negative, border case' scenarios, and I should have run through some games with a 'mature' starting condition – testing whether the railroads reported their correct costs by starting the game with a player owning multiple, or testing whether houses can be built and sold in proper order by giving players monopolies on their first turn.

INDIVIDUAL RATING

In addition to knowing how well the game development process went, it's important to know how well I did on a personal level.

ATTENDANCE

I was very committed to attendance, but didn't put in enough out-of-class time. There are many areas which would have been greatly improved by two or three extra hours of project work per week – in particular, UI and asset design, and testing and stability. Showing up to class, and making as full a use of classtime as possible, was not my primary challenge for this semester.

SCRIPTING ABILITY AND OUTPUT

I was able to navigate Vicious Engine's scripting environment fairly well, especially given its level of documentation. For the most part, given the parts of the engine on which we were instructed, I was able to implement a very rich feature set; unfortunately, between a shaky UI and a lack of key systems, much of the scripting is not directly demonstratable.

CLASS INTERACTION

I attempted to help as many class members as I could, but I can tend to overstep and focus on solving the problem rather than helping solve the problem. Still, I attempted to respond promptly and accurately any time someone asked for help.

WORK

I was very dedicated to working on my project while in class. I feel I got an adequately large amount of the project complete. Had I prioritized the project higher in non-class time, it's likely I would have gotten it to the point of polishing and feature creep (which is exactly where you want to be when a deadline comes around).

QUALITY OF GAME

I am very happy with the overall quality of the game, given the short schedule and lack of knowledge about the engine. There are some places where I could have polished more, or established even a prototype/placeholder operation – in particular, an AI state machine could have been very impressive, and even a simple property trading mechanism, however far from my original concepts for player interaction, would have allowed me to better demonstrate the property improvement system. In short, it makes a good, quick demo, but doesn't fit the requirements of an MVP (minimally-viable product).

PRODUCTION SUMMARY

Doing a true milestone breakdown would be difficult, because the actual sequence of development differed radically from the written milestone plan.

Originally, I had planned to implement the core gameplay screens – players, board/jail, and assets – along with basic hotseat gameplay. This milestone was never fully achieved; we first built out the splash screens and main menu, and then proceeded to an in-game screen. According to the class milestones, even working on the player interaction screen would have been ungraded (wasted) effort.

The next planned milestone was for tracking property state. This was implemented, but much later in the actual project (in fact, in terms of class milestones this was one of the last requirements). As it turns out, tracking any single property state is as easy as tracking all possible property states – the biggest challenge was in building the interface which allowed you to have an effect on these states.

The third milestone was to be transactions and auctions – interactions between players which interrupted normal turn order. This, to me, was the biggest missing milestone of all of them; if I had been able to implement this, it would be possible to actually play a game of Monopoly hotseat style. As mentioned, however, these features weren't even a part of the class milestones, so in this case there is no permanent damage done.

The fourth milestone would have been to implement the cards and special case spaces (like taxes). These were done, largely because once I saw how they could be handled, it wasn't much work to slot them in. The payoff for this was moderate, but probably worth the small amount of effort.

The fifth milestone would be to handle the 'In Jail' state; however, this was addressed fairly early in the class milestone schedule, so this feature was moved forward. This turned out to be one the buggiest parts of the final product, but at least the features were there in spirit.

The sixth milestone was to add the 'fluff' screens – a main menu and some options. As noted above, these were the first thing instructed, and also an important part of the first class milestone, so they were completed very early in the project.

Finally, a simple AI was to be implemented. The basic strategy for winning Monopoly would have been easy to implement – buy everything you land on until you can't buy any more. Some more advanced strategies would require some statistical analysis – such as which properties are worth how much based on how often they are landed on, and what percentage of your cash assets should be used in an auction for a property based on your level of investment in that group at that time. These features, however, would be likely to go unnoticed by all but the most adroit players, making it an excellent set of features for a post-release patch. Either way, there was nothing about AI in the class milestones, so all this effort would have been 'wasted' anyway.

ASSESSMENT OF FINAL GAME

The final game didn't turn out too poorly, given the schedule and resources. It could definitely have used the influence of another person or two – specifically someone with some artistic training.

DESIGN

For the majority of the project, I feel the design was very successful. I attempted to 'flatten out' the control scheme, since navigating a deep menu system is my biggest pet peeve for console and handheld gaming; had I been able to properly convey this principle and still meet experienced player expectations, if would have been very successful. As it was, some parts were clunky for a mouse to operate because the screens were designed with the idea of a moveable selector in mind.

ART

This was, obviously, the greatest flaw in the final game. Most assets I produced or used didn't really take the realities of the platform into account – specifically, a tiny, low-res screen. In addition, even the placeholders I used were barely adequate; the worked as indicators of the facts of the game, but did very little to convey the feeling. There were also missed opportunities to use pictures instead of words, such as for intra-screen navigation and the dice rolls.

PROGRAMMING

The logic of the final product was, for the most part, both thorough and sound. There are a couple places that needed actual testing, and some features had been implemented but never actually tried, but for the most part the rules were executed accurately in code. In addition, the game objects were structured to allow for the further development of the codebase into things like AI and player interactions. Overall, despite my griping about the style of Vicious Engine's scripting, I feel I was able to pull off a real project.

PRODUCTION

In general, the production side of things – keeping track of resources, scheduling, implemented and missing features, and the like – was fairly weak. On the other hand, the final product would have very little mystery – the game is well established, and the rules are very clear, so little playability testing or tracking is required. As such, I would rate the overall production of the game to be adequate – it neither hindered nor helped the quality and schedule of the final product.

NEW MILESTONES

If given the chance to finish the game completely, these are the milestones I would work toward:

1. Completely debug all the existing code. This would include all the actions of the cards, and the in-jail interface. At this point, there's no use building more code when what you have doesn't work.
2. Refactor the existing code. Specifically, there are a few actions common to every player on every turn, both human and AI, which could stand to be turned into functions for clarity and stability; in particular, moving the player tokens around. Far too much code was copied and pasted, rather than packaged for better reuse.
3. Finish out player interaction. Even the simple case of trading properties, or selling from one player to another, would magnify the completeness of the game. Technically, auctions are a critical part of the Monopoly rules, but it's rare that people have actually played the game by the rules. In this case, I feel it would be better to err on the side of familiarity than accuracy.
4. Once the player interaction screen is done, I'd put in place the ability to request the sale of a property from another player, and an auction screen. Both of these actions can be handled player-to-player in an actual hotseat game, but it's better to codify it all for the next step.
5. The next step is simple AI so that a hotseat game isn't required. All it would have to do is bid on auctions, and buy properties it can afford when it lands on them.
6. After all that's in place, I feel the actual game has been implemented, so now it's time for polish – at this point, I'd redesign the UI to fit the features which exist and the ways in which things interact. After this, I'd call the game a minimum viable product.
7. Finally, the icing on the cake – deeper AI. Something with strategies to get high-value properties, something which will play the player in an auction or transaction (raise the bid amounts based on the opponent's financial position in order to sap their resources), and carefully plan how to get monopolies without the players realizing what's going on.
8. Given time and resource, the final step would be true multiplayer – meaning, multiple devices engaged in the same game. Many of the Monopoly rules call out player interactions and their limits outside of a context of being the player with the dice, so a true interpretation would allow players to build and buy even when it's not their turn.