Projects and Portfolios — An Educator's Reflections on the Summative Assessments in a Game Programming Course

Steffan Hooper
University of Auckland
New Zealand
steffan.hooper@auckland.ac.nz

Burkhard Wünsche
University of Auckland
New Zealand
burkhard@cs.auckland.ac.nz

Stefan Marks
Auckland University of Technology
New Zealand
stefan.marks@aut.ac.nz

Andrew Luxton-Reilly
University of Auckland
New Zealand
a.luxton-reilly@auckland.ac.nz

Paul Denny
University of Auckland
New Zealand
paul@cs.auckland.ac.nz

Abstract

This paper describes the assessments in a third-year undergraduate *Game Programming* elective course. Three summative assessments are conducted throughout the semester: an Individual Game Project, a Team Game Project, and a Personal Portfolio. Throughout the course, learners develop a variety of formative artefacts that build towards their individual game, and then are further extended collaboratively to develop their team game. Artefacts generated by learners are also added to their Personal Portfolio, which is designed to target game industry employment opportunities. We provide critical reflection on the implementation of the assessments, and discuss some of the challenges. We share our insights to assist educators wanting to explore authentic game industry-style assessment.

CCS Concepts

• **Applied computing** → **Computer games**; • **Computing methodologies** → *Computer graphics*; • **Social and professional topics** → **Computing education**.

Keywords

Assessment, game development projects, game programming, portfolio, teamwork, undergraduate

ACM Reference Format:
Steffan Hooper, Burkhard Wünsche, Stefan Marks, Andrew Luxton-Reilly, and Paul Denny. 2024. Projects and Portfolios — An Educator's Reflections on the Summative Assessments in a Game Programming Course. In *SIGGRAPH Asia 2024 Educator's Forum (SA Educator's Forum '24), December 03–06, 2024, Tokyo, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3680533.3697071

1 Introduction

Skilled game programmers are in demand [IGEA 2023; Kenwright 2016; NZGDA 2022]. To train undergraduate computing students in the specialised programming techniques used to develop games, we offered a third-year *Game Programming* elective course. Its primary goal was to engage learners with a practical foundation using various contemporary game development tools and programming libraries. A constructivist approach was taken, where learning was realised through doing or making [Nikolov 2023; Wu and Wang 2012]. At the beginning of the course, learners used small formative game programming exercises to practice newly learnt techniques. Then summative project-based assignments required students to creatively develop video games, which were used to assess learner competency against the course's learning outcomes. The course also emphasised team-based game projects that were collaboratively designed, developed, and managed, in a way that simulated a game studio — putting theory into practice — but within the constraints of a university setting, such as having limited directed contact time.

The project assessments' core focus was the creation of playable games. Learners were free to choose the type of game and topic for their projects — a strategy to help learner motivation and interest [Taylor and Baskett 2009]. By constructing games that were personally meaningful to learners, learning could become active and self-directed [Wu and Wang 2012]. The assessments were open-ended, empowering learners to scope their games to their own desires, skills, and available timeframes, with high-level negotiation and instructor advice along the way. The game designs were not necessarily new ideas. Learners were encouraged to clone existing fun game ideas, or create their own if they were inspired to do so. The important outcomes were that they were successful in creating a playable game, as well as productively collaborating, while programming significant features for the benefit of their team, and then also critically reflecting upon their lessons learnt from each project's successes and failures.

The course mimicked industry practices, where the game development production life-cycle was followed, initially as inspired by Bethke [2003] and Irish [2005], with a *Pre-production* design and planning phase, a *Production* development phase, and a *Post-production* reflection phase, adapted for an educational setting.

In the following sections, we provide educator insights into the overall use of formative exercises, the design and implementation of the summative assessments, and the associated strategies and supporting technology used within the course — both for individual and team-based assignments. We conclude with educator reflections and discussion of using this assessment strategy.

SA Educator's Forum '24, December 03–06, 2024, Tokyo, Japan

ACM ISBN 979-8-4007-1136-7/24/12
https://doi.org/10.1145/3680533.3697071

2 Related Work

Game programming courses can be assessed in different ways. For example, Peng [2015] describes an introductory game programming course using a combination of many assessment types, including attendance, quizzes, a mid-term exam, three projects and a final exam. Amador and Gomes [2016] present an advanced Masters-level game engine development course that assessed learners using a project, a presentation based upon a scientific paper, three written theoretical tests and six engine programming tests. However, some game development courses use project-based learning [Mateas and Whitehead 2007; Parberry et al. 2005], with no final exam [Jones 2000; Ritzhaupt 2009]. Assessments can include individual and collaborative teamwork, with reported team sizes varying from just two members [Peng 2015], three to five members [Stephenson et al. 2016], or bigger teams of seven or more students [Zagal and Sharp 2011]. Sometimes, the formation of teams is self-organising, and they can also be multidisciplinary [Sturtevant et al. 2008].

Some game development educators have encouraged the construction of websites or portfolios to showcase student work. Kenwright [2016] describes a degree programme that runs a workshop on creating a website or portfolio, and Argent et al. [2006] describe their game development degree's capstone course where students work with instructors to assemble a portfolio, while completing their senior project. Similar to our course, Munkvold's [2017] course is assessed with a portfolio, assembled from work logs, documented work, and a game — with learners undertaking game development in a simulated studio environment.

Courses have included the creation of game pitch and design documents to help plan game projects [Peng 2015; Rankin et al. 2007; Ritzhaupt 2009; Sturtevant et al. 2008]. Development milestones have been used to create project deadlines [Gestwicki and Sun 2008; Stephenson et al. 2016; Sturtevant et al. 2008], and iterative development practices undertaken [Rankin et al. 2007]. Prior works also provide advice for game programming educators on how to prepare a playable game and conduct postmortem reflection [Parberry et al. 2006], as well as the important consideration of provisioning game programming learners with game art assets [Parberry et al. 2005].

Additionally, for further perspectives on game development assessments in computing education, introductory CS1 [Luxton-Reilly and Denny 2009], CS2 [Black 2016; Lewis and Massingill 2006], and Computer Graphics [Suselo et al. 2017] courses have incorporated aspects of game programming within their assignments, and educators have explored through interviews what makes a good game programming assignment for CS1 courses [Cliburn and Miller 2008].

In this paper, we describe and reflect upon the three assessments used within a third-year game programming course, comprising an individually crafted game, a collaborative team-developed game, and a portfolio that collates a learner's formative and summative work throughout the semester. We report our experiences using the combination of these assessments within a single elective course.

3 Course Overview

The course was delivered over a 12-week semester, with two 2-hour directed class sessions per week taught by a single instructor. Learners were also expected to engage in six to eight hours of self-directed course activities per week. Early in the course schedule, class sessions included short lecture-style delivery, live game programming demonstrations, and practical formative game programming exercise activities. Later in the semester, classes were solely dedicated to team-based project development activities. Formative exercises were used to scaffold learning new skills and knowledge and to work towards creating practical applications within game scenarios. These formative exercises helped to build reusable components for the subsequent summative assessment game projects.

3.1 Summative Assessment Overview

The assessment structure comprised two individual assessments and a group-based assessment, as shown in Table 1. The course also had an overall requirement for the learner to score a minimum mark of 35% in each assessment and an overall result of at least 50% to pass. Each assessment had a PDF assignment brief, distributed via the university's Learning Management System (LMS), that described the assessment requirements, and provided detailed marking criteria divided into relevant criterion with grade-banded performance descriptions for A, B, C, and D characteristics.

Table 1: Course assessment structure

Assessment Event	Weighting	Mode	Week Issued	Week Due
Personal Portfolio	15%	Individual	1	13
Individual Game Project	35%	Individual	4	8
Team Game Project	50%	Group	6	13

The high-level instructions for each assessment were as follows:

3.1.1 Personal Portfolio — "Throughout the semester, curate a personal portfolio of game programming artefacts that reflect your learning and development. The portfolio's target audience should be a prospective game development employer."

3.1.2 Individual Game Project (IGP) — "Individually design and develop a real-time interactive video game in C++ and Visual Studio Enterprise 2017, utilising techniques and technologies presented in class. The game must showcase game programming aspects learnt during studio sessions and has the scope to include additional unfamiliar, and sometimes complex, game programming solutions."

3.1.3 Team Game Project (TGP) — "In a team of four or five members collaborate to design, plan, schedule, and develop a video game. Individuals must share accountability for the overall game production and delivery while also taking responsibility for programming various and significant aspects of the overall game."

3.2 Monitoring Assessment Progress

Each of the assessments had regular instructor check-ins throughout the semester to encourage incremental work — attempting to circumvent the potential for learners to try to develop an entire game, or portfolio, the night before the deadline. Importantly, these reviews provided a chance for individualised, work-in-progress feedback from the instructor and advice regarding future development steps, risk monitoring, and problem resolution. Some check-ins were scheduled as formative, in-class reviews, with non-graded

one-to-one feedback. In contrast, other check-ins occurred via delivery of artefacts to the LMS at scheduled project milestones. These artefacts were assessed against the associated assignment brief's marking rubric, with the result contributing towards part of the overall grade for the assessment.

Version control using Subversion (SVN)[1] was also utilised. Learners could commit their source code and associated game project artefacts while they developed each game. A private, individualised directory was available to each learner, and the instructor could also view the area. The SVN server used university-provisioned hardware, and user access was administered by the instructor. Each team was allocated a shared directory for their collaborative project for teamwork. Individuals could generate a text log detailing their commits to SVN as part of the evidence submitted for their TGP. Using SVN enabled the instructor to monitor incremental progress as the variety of learner game projects developed.

3.3 Formative Exercises

Short formative exercises were completed in class and also during self-directed time. The course contained a variety of exercises, and learners could choose to complete them as needed. Formative exercises were not marked. However, the artefacts generated could be added to a learner's *Personal Portfolio*. These artefacts would demonstrate learning practices and upskilling results, often with small tech-demo creations. Learners could also include short reflections on their knowledge and skills growth and document the problem-solving processes they had employed.

Experimenting with the formative exercises helped accelerate building features for the subsequent IGP and TGP assessments. More detail on this aspect of the course can be found in [Hooper et al. 2024b]. Generally, these exercises could be classified into broad categories of: *C++* — language upskilling and feature implementation; *Tools Development* — programming tasks that result in useful applications that accelerate game development-related activities; and *Game Clone* — development of specific game scenarios to highlight particular game programming techniques relevant to a certain topic or lesson within the course.

3.3.1 Selected formative exercise examples:

- *Tools Development:* **`cpp` and `h` Generator** — Create a program that outputs `.cpp` and `.h` files for a new class based upon a user-specified name, following the course-established initial C++ file layout, including stubs for constructor, destructor, and game loop methods such as `Process` and `Draw`;
- *C++:* **Start-up Splash Screen** — Implement sprite asset loading and rendering with alpha-blended fade animation;
- *C++:* **Bouncing Balls: N-Squared versus Quadtree** — Implement moving 2D ball sprites with collision detection and response, bouncing off the edge of the game window such that they are always fully visible. Any overlapping balls are to be coloured red, and initially, overlaps are detected using an n-squared approach. This is then improved by implementing a quadtree to minimise collision checks;
- *C++:* **FMOD[2] Middleware Integration** — Audio application programming interface (API) integration, following API guidelines, including logo usage requirements, reviewing the "Getting Started" documentation, configuring project settings for header and library dependencies. Implement a sound asset resource manager.
- *Game Clone:* **Space Invaders [Taito 1978] clone** — Build the classic arcade game where the player controls a horizontally moving laser cannon, attempting to defeat wave after wave of descending alien invaders.

3.3.2 Build Targets. Formative exercises established the *Debug* and *Release* build target pattern, following good game programming practices inspired by McShaffry and Graham [2012]. Summative IGP and TGP requirements expected learners to deliver the two build target executables in a functioning state, where the game could successfully build and execute from either target. Projects were configured to launch from a single *Working Directory* that contained game-ready assets. The final build for each project was delivered with both Release and Debug build executables alongside the required asset files.

3.4 Game Art Assets — Programmer-art

Learners often used open source art assets, such as those available from the *Open Game Art* website[3], but sometimes assets were personally crafted when a learner had pre-existing artistic skills that they wanted to leverage when creating their game projects. Using any third-party-created assets required the learner to document where they had obtained the assets and that they had the rights to use them within their game. This was done via text documentation, generally a `readme` file that accompanied the final game build submission, and within in-game credits. Encouraging accurate crediting practices also created an opportunity to have learners engage with the industry-relevant International Game Developers Association (IGDA) *Game Crediting Policies* [IGDA 2014, 2023] alongside appropriate academic referencing practices.

4 Summative Assessments

4.1 Personal Portfolio

The *Personal Portfolio* tasked learners with crafting a portfolio of their work, including artefacts such as their summative game projects, and their formative exercises. The collation of this work empowered learners to share their personal game programming story, showcasing their growth in skills and knowledge, and their reflections throughout the semester. There was no prescribed requirement for the portfolio to be a website or live online. However, some learners chose to make their portfolios publicly available (see Figure 1).

The assignment brief suggested possible structures, including section ideas, such as *Home/Overview*, *Games*, *Tech Demos*, *Projects*, *Side-projects*, *Resume/CV*, *Blog*, and *Contact Links*, as well as ideas for the types of artefacts that could be incorporated, such as images, videos, source code, technical designs, reflections, and downloadable/playable games. The brief also asked learners to ensure adequate referencing was used when referring to other people's work, especially considering the portfolio was likely to share details

[1] https://subversion.apache.org/
[2] https://www.fmod.com/
[3] https://opengameart.org/

of the group-developed TGP. Two in-class formative reviews occurred during the semester in weeks 6 and 10, where the instructor could critique the portfolio work-in-progress, and provide further motivation or ideas for portfolio improvement.

Example real-world game industry portfolios such as Hannam [2024] and Ponichtera [2024] were shared with learners at the start of the course. These examples showcased the variety of game programming portfolio possibilities. Additionally, a variety of archived portfolio-related game industry conference presentations from the *GDC Vault* website[4] were shared with learners for further inspiration and real-world expectation-setting. Ideally, by the end of the course, the portfolio could be given to a prospective employer in the game industry, for example, when applying for a job.

4.1.1 Marking Criteria: There were three marking criteria, with associated grade band descriptions focused on the following aspects:

- *Artefacts (60%)* — encompassing: Selection, Range, Purpose, Complexity, Media, Description, and Reflection;
- *Presentation (20%)* — encompassing: Usability, Accessibility, Organisation, Navigation, and Functionality;
- *Writing Standards (20%)* — encompassing: Spelling, Grammar, Referencing, and Attribution.

4.2 Individual Game Project (IGP)

The IGP asked learners to individually create a game. Learners were expected to deliver source code, all assets required to build the game, and a pre-built, final *Gold Milestone* Release build executable. The assignment brief outlined feature requirements for learners to aim to develop, including a variety of technical features to realise in their game design using C++, with appropriate source and header file layouts, and associated object-oriented programming techniques. The work was required to demonstrate a real-time interactive game simulation, including: animated sprites with scaling, rotation, layering effects, and alpha or colour blending; particle systems; text rendering. Additionally, the game's design scope had to incorporate features to demonstrate game programming techniques covering audio, artificial intelligence, data-driven design, middleware and library usage, testing and debugging features, and audit trail logging. The interface features required that the game have player input via a game controller, and for the game to start fullscreen with appropriate splash screens, and that user instructions and credits were present in-game, as well for the player to be able to restart the game without exiting the executable.

A pre-production planning phase was undertaken, resulting in a short *Game Pitch* — a single-page, non-technical document that conveyed the core purpose of the game, including title, genre, target audience, gameplay and mechanics, player goals, key/core features, user interface design overview, and unique selling points. Further to this concept proposal, students had to provide a more thorough *Game Design Document* including the game's rules, mechanics, specific feature decisions, and key algorithms that govern gameplay, as well as details for the proposed control scheme, mock-up interface, and sample screen layouts for each stage of the game.

Learners were encouraged to include cheat features that could assist development, game-tuning, and a proposed required assets

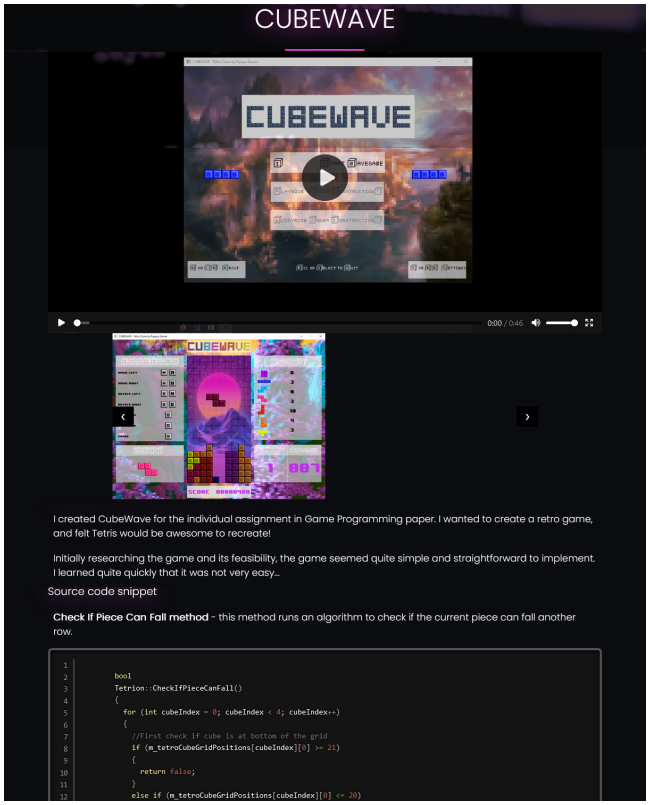


Figure 1: Example webpage screenshot from a portfolio featuring a game play-through video capture, in-game screenshots, a short reflection statement, and a game source code snippet [Ashizumi-Munn 2023]

list (e.g., sprites, animations, audio). Alongside their game design, students prepared a *Technical Design Document* that included descriptions of the logic and technical algorithms required to implement the game features. These could be described using simple flowcharts, or UML[5] Activity, Sequence, State and Class diagrams — showing proposed object hierarchy and interaction, with key properties and functionality to be implemented. Learners were also encouraged to include any designs for technical debug features, their proposed coding standard and naming schemes, and relevant file formats. Also expected was an acceptance test plan, containing ten questions regarding game functionality, as a basic test suite that could be used to validate the game's completeness via user or play testing. Each document guided the vision and the production of the proposed game. Learners were also required to include a bibliography in the university's standard *APA 7th edition*-style[6].

Throughout the production phase, while the game was being developed, learners were encouraged to make regular and well-commented commits to SVN as they worked on the project. Two milestone deliverables to be submitted via the LMS were defined for the assignment: a *First-Walk-and-Talk* in-class demo; and the final *Gold* milestone. The assignment brief specified expectations for

[4]https://gdcvault.com/

[5]https://www.uml.org/

[6]https://apastyle.apa.org/

Build and Runtime Quality, with the project required to successfully build with *no warnings or errors at Warning Level 3*, and be free of memory leaks, runtime bugs and crashes. The brief also provided a sample of the expected submission file structure, including the folders: `docs`, `game`, `lib`, `source`, `test` — as inspired by McShaffry and Graham [2012].

In the post-production phase, learners reflected on their project by writing a postmortem in the style of a commercial game industry postmortem[7], including five sections: Overview, What Went Right, What Went Wrong, Lessons Learnt, and Conclusion. A `readme` document was also created that detailed the game's control scheme and the authorship details.

4.2.1 Marking Criteria: There were five marking criteria, with associated grade band descriptions focused on the following aspects:

- *Design (12.5%)* — encompassing: Game Pitch, Game Design Document, and Technical Design Document;
- *Gold Game Implementation (25%)* — encompassing: Game play, Interaction, Input, Game restarting, Bugs, and Crashes;
- *Gold Graphics Implementation (25%)* — encompassing: Graphical Effects, Sprites, Particles, and Animation;
- *Gold Milestone Build Quality (25%)* — encompassing: Compilation, Overall game, Technical features, Complexity, and Middleware usage;
- *Reporting (12.5%)* — encompassing: SVN usage, First-Walk-and-Talk Milestone, and Postmortem.

4.3 Team Game Project (TGP)

Teams were self-forming, promoting learner agency, where groups would generally coalesce around a popular or interesting game concept. There were some constraints imposed, such as possible team size, ideally four to five members, and with the provision for the instructor to rearrange team makeup if not all individuals in the cohort were accommodated for by the team formation deadline. The assignment brief instructed learners to develop team goals and overall project milestones — these were learner-driven, but with room for instructor input and negotiation. Each team member was responsible for programming particular aspects of the game, with a shared group responsibility for the overall game delivery.

For the TGP, in the pre-production phase, self-described team milestones, with target dates and required associated game features were established by the learners. The assignment did expect teams to establish targets for milestones: *First-Walk-and-Talk* with key game features demonstratable, an *Alpha* milestone where the vast majority of game features had been implemented, a *Beta* milestone where the game would be feature-complete, but perhaps still have some minor bugs present — in the style of game industry bug classifications inspired by Levy and Novak [2009], and a final *Gold* build that was the feature-complete game. The deadline for the Gold milestone was specified in the assignment brief, coinciding with the end of the semester's teaching time.

The planning and scoping documents similar to the IGP were also required, with additional potential for more elaborate aspects to be created within the design document, such as proposed level or mission design, comprehensive asset list requirements including 2D and 3D art, animation, sound, music, voice script, etc. The technical design could be more expansive but was also required to consider an appropriate development methodology to facilitate team-based work, risk identification, feature importance and scoping. To encourage buy-in, and agreement to the proposed development directions, a *Team Sign-off* section was also required in each document, where learners signed to show their agreement with the plans described in the pre-production artefacts. The resulting artefacts were submitted via a group submission to the LMS.

For the production phase, learners were encouraged to keep an individual *Developer Journal*, which aggregated their personal notes throughout the project and detailed their activities. This could include their own record of team meetings, short summaries of team discussions, and the resulting important decisions made, e.g., action points, as well as any further research and design activity undertaken and resulting findings. This was particularly useful when additional technical designs beyond the initial pre-production phase were required, enabling further planning activity to occur and be clearly documented and attributed to the individual responsible. The journals were submitted alongside the Gold milestone group submission to the LMS, ensuring transparency in claimed responsibility for features developed and activities undertaken by each team member.

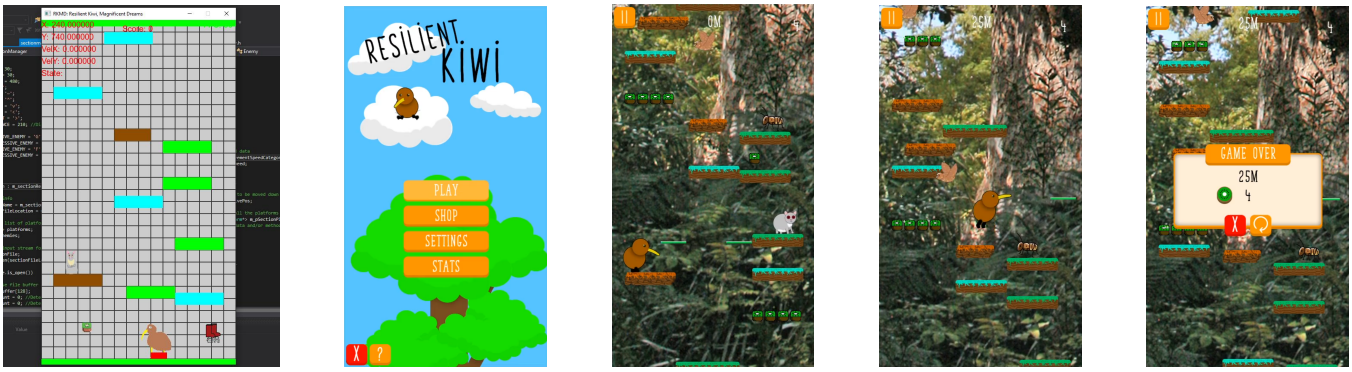
Work-in-progress check-ins and instructor feedback occurred via *Weekly Delta Demonstrations*, adapted from, and inspired by Irish's *Daily Delta Reports* [2005], where team members would quickly demonstrate, and discuss with the instructor, the new features they had been working on, while highlighting any areas of concern where they may need further assistance. The brief also allowed learners to request any additional middleware or third-party code to the instructor via an LMS *Discussions* forum post. This kept requests and the resulting discussion and approval transparent and open to the entire cohort. It empowered competing teams to leverage approved middleware within their projects and be aware of the possible implications, keeping a level playing field for all teams. Learners were encouraged to do due diligence, evaluate their proposed technology's suitability, and conduct a risk analysis. This request process helped avoid possible problems unforeseen by learners, leveraging the instructor's expertise to avoid any catastrophic scenarios.

The resulting TGP was packaged and delivered as a *Gold Milestone* submission (for an example, see Figure 2), including full source code and assets, `readme` document, *Environment Setup and Build* instructions, a *Known Issues* document, task tracking and milestone completion details based upon the team's chosen development methodology, and a copy of each individual's SVN log as a text file.

Finally, during post-production, individuals wrote their postmortem reflections. Collectively, each team delivered a presentation of their game to the cohort. A final play session occurred at the end of the course — the course's version of a showcase [Gestwicki and Sun 2008; Ritzhaupt 2009]. This was scheduled in the last class session, after the *Beta*, but before the *Gold* milestone.

4.3.1 Marking Criteria: There were four marking criteria, each with either a shared group-based result, or individual-based result, and with associated grade band descriptions focused on the following aspects:

[7] https://archive.gamedev.net/archive/reference/articles/article977.html



(a) Work-in-progress build (b) Title screen menu (c) In-game capture (d) Jumping kiwi (e) Game Over summary

Figure 2: Example screenshots from a *Team Game Project* [Ashizumi-Munn 2023]

- *Pre-production Planning (Group-based: 15%)* — encompassing: Team formation, Game Pitch, Game Design Document, Technical Design Document, Scheduling, and Milestones;
- *Weekly Delta Demonstrations (Individual-based: 15%)* — encompassing: Functionality, Gameplay, Implementation, Complexity, Progression, Attendance, and Week 12 Beta demo;
- *Gold Build (Individual-based: 15%)* — encompassing: Builds, Playable, Known Issues, `readme` document, Environment Setup and Build Instructions, and Gold Milestone Report;
- *Individual Contribution (Individual-based: 55%)* — encompassing: Features developed, Implementation, Complexity, Studio practice, Attendance, Contribution, Developer Journal, SVN commits, and Postmortem.

5 Discussion and Reflections

The "Computer Science Curricula 2023" guidelines describe professional dispositions for the *computer graphics and interactive techniques* knowledge area [Kumar and Raj 2024; Kumar et al. 2024]. In line with these guidelines, our course assessments engaged learners with self-directed and life-long learning, and encouraged critical self-reflection on the work they produced. During pre-production, learners researched and designed technical solutions, with project planning activities that prepare for upcoming development tasks. IGP artefacts created can be expanded upon and refined in the subsequent TGP, resulting in experiences that can be built upon beyond course completion. Furthermore, the assessments encourage collaborative behaviours between learners, where they engage with teamwork as game programmers, and regularly seek and provide feedback within their teams (and cohort) as development occurs.

Learners demonstrated their growing professional communication behaviours in written form with the various design, reflective and reporting documents, and verbally through presenting their work-in-progress, sharing details of their contributions with the instructor via the weekly delta demos and formative reviews, and within their teams. Programmer-centric communication practices were also evidenced through version control practices, with commit messages, and the ability to coherently merge individual programming tasks with the team's evolving codebase. The final game presentations exemplified the completed games, and the creative work undertaken, alongside critical learner reflections.

Learners were empowered to lead their own game programming journey, with the agency to decide on the direction of their game projects, and to regularly reflect on their progress. Lessons learnt from the IGP were valuable for the subsequent TGP, and the reuse and further development of source code that had been created individually, benefited the team-based projects. Teams could pick the best components of different individuals' IGP implementations to craft a starting point for their collaborative development efforts. The IGP and TGP briefs were designed to mimic real-world game development practices, not only aiming to create playable games, but also to generate useful artefacts, such as planning documents and postmortems, release build packaging, build instructions, and the documenting of known issues — demonstrating good practices, transparent project monitoring, and risk management strategies. Each assessment used industry-relevant vocabulary, immersing learners in the game development world.

The transition from completing their IGP into designing a TGP, where learners finished working on their individual game, reflected upon it, and then began working in a team environment, was a compelling step. Learners would take lessons learnt, and prior source code experience, into the collaborative team workspace. Students were invigorated by the fresh start. Team foundations could be formed by sharing their IGP with their peers, and learners could gauge a peer's potential skills, and work attitudes, that were available to potentially collaborate with.

The game pitches were instrumental in focusing teams on a single game idea. Having written a pitch for their IGP, learners had a good idea of what was required to pitch a game idea to their peers and then collectively refine an idea into a single-page pitch that all members could agree to develop further. Crafting a game design document helped to hone game ideas, allowing teams to thrash out the game concept and its possible implementation needs more clearly. Incremental levels of scope could be defined, with *must-have*, *nice-to-have*, and *wish-list* game features. This also overlapped into the technical design document creation, with teams compiling relevant technical resources into a centralised document. Individuals had clear areas of responsibility to research,

design and develop for — but it also helped to convey ideas between team members and provide a shared understanding of development areas and game features. Individuals could establish ownership of specific features or topic areas on which they wanted to work within the game production and take on leadership of technical areas, further developing their own expertise and sharing this with their peers. Learners could assess project risks and validate peer choices related to potential development directions, however, the student-focused Kirk et al. [2024] risk framework could be used to further enhance project risk management within the course.

Regular formative check-ins helped motivate learners to continually be productive in working on each assessment. However, there were instances when learners made late changes to their projects, sometimes radical decisions that drastically impacted their games. Generally, the earlier the formative feedback, the more helpful it could be in shaping the project and circumventing problems. The instructor had limited opportunity to help learners outside of class. At times, help was sought too late, e.g., right before the deadline.

The TGP postmortem enabled confidential reporting. Interpersonal dynamics could be safely disclosed, with the potential to discuss better ways to handle dysfunction in the future. Marking TGP group work could be challenging — the criteria split elements between items that could be assessed individually and items that were distinctly team crafted. The criteria were mainly comprised of individual aspects, which aimed to reward individuals who made useful contributions, while avoiding rewarding individuals who contributed little (see Section 4.3.1). Challenges included dissecting the project, and its artefacts, and comparing it against the varied task completion reporting from each team. Sometimes task tracking would be poorly handled. It is important to note that to scale the course to support a large number of learners, more resources would be required. Similar to a lead/senior programmer in the game industry, one instructor can only mentor so many learners and projects before the workload becomes unmanageable. A key consideration was that each assessment required manual human marking, with multiple feedback points to keep learners on track and motivated. Automated tools may be able to help address these issues [Hooper et al. 2024a; Luxton-Reilly et al. 2023].

SVN was a critical tool for the course. There were multiple benefits: learner work was backed up to a university server, and progress could be easily transferred between on-campus workstations and personal home-based devices. A history of commits provided an audit trail of who contributed what and when, clearly showing feature development history — the instructor could also see each individual progressively working. The *TortoiseSVN*[8] interface for committing and merging was often found to be simple for learners to use — at times, they contrasted it with their prior experiences with other version control systems. Each cohort worked within a fresh repository, with all work done in one branch, simplifying learners' update-merge-commit cycle. Assisting learners with the investigation and resolution of bugs was aided by the use of SVN. Learners could commit problematic code, and provide the instructor with the revision number containing their particular problem. The instructor could then investigate and debug, either directly committing a fix, or documenting a guide to fix the issue within the codebase. When the instructor worked on a learner's workstation to assist with a repair, the fix could be committed with a comment containing the instructor's initials — attributing the educator's contributions to a learner's artefacts within the SVN log.

The *Personal Portfolio* was valuable for creating a place for learners to share their growth and the variety of small tech demo artefacts developed throughout the course. A dedicated place to store and present the formative exercise outputs helped motivate the need to work on the exercises. The portfolio also taught learners how to best present their work to third-parties. For example, how to capture good quality video content, paying attention to aspects that may seem minor, but can be distracting to the perception of overall quality — such as, avoiding recording the mouse cursor, or cutting a video to be a short, focused clip that makes it snappy and eye-catching for the viewer. Being able to present and highlight a game feature that may have been a significant technical effort to develop, but perhaps would otherwise be overlooked within the overall game, was empowering. The resulting portfolios could also endure beyond the end of the course — learners would use them immediately for job applications and continue building them into their professional careers.

Workload and learner attention could present a challenge at times. Learners were generally enrolled in a full course load of four courses concurrently, so their focus and efforts were split across multiple subject areas. One of these was a final capstone course, which seemed to be a priority for learners. However, learners also expressed their enjoyment of learning about, and creating, technical game programming solutions and the course and its assessments appeared to be the highlight of their undergraduate experience. To capitalise on the students' positive experiences, while also providing a potential solution to workload concerns, the credit value for the course could be increased to give learners more directed contact hours and further opportunity for instructor assistance.

5.1 Aggregated End-of-Course Evaluations

Anonymous feedback was collected from learners centrally by the university, although not for every year that the course was run. Aggregated responses were made available to the instructor once the course and final results were completed — a detailed description of this course evaluation process can be found in [Marks and Gil Parga 2023]. Feedback was generally positive regarding the appropriateness of the course assessment (see Table 2). Additionally, learners could provide anonymous written feedback comments for the *Assessments were a fair measure of my learning* metric, typical examples of which were: *"They definitely pushed me to my best."*; *"Assignment details are clear and precise"*; *"Extremely detailed marking schedules."* — highlighting learner satisfaction with the assessment instructions, marking criteria, and project motivation.

Table 2: End-of-course evaluations, *Satisfied* result for assessments

Year:	**2016:**	**2017:**	**2019:**	**2020:**
Appropriate Assessment:	87.5%	80.0%	88.9%	100.0%

[8] https://tortoisesvn.net/

6 Conclusion

The *Game Programming* course and its assessments as described in this paper mimicked industry practices, following industry-style production phases, engaging learners in teamwork, effectively using version control, regular game testing, and preparing a portfolio — all in a simulated studio environment. Overall, we believe that the assessment approach added significant value to the learners' experience and we encourage educators to consider adopting a portfolio to supplement more traditional assessments.

Acknowledgments

We thank Auckland University of Technology for offering the course — *COMP710 Game Programming* — and all the COMP710 students.

References

Gonçalo Amador and Abel Gomes. 2016. A Video Games Technologies Course: Teaching, Learning, and Research. In *EG 2016 - Education Papers*. The Eurographics Association. https://doi.org/10.2312/eged.20161027

Lawrence Argent, Bill Depper, Rafael Fajardo, Sarah Gjertson, Scott Leutenegger, Mario Lopez, and Jeff Rutenbeck. 2006. Building a Game Development Program. *Computer* 39, 6 (2006), 52–60. https://doi.org/10.1109/MC.2006.189

Maya Ashizumi-Munn. 2023. //mayapapaya. https://7473fb9c-ffea-4e02-a3be-449c13ea2cc3.repl.co/. [Accessed 22-Sept-2023].

Erik Bethke. 2003. *Game Development and Production*. Wordware Publishing, Inc.

Michael David Black. 2016. Seven Semesters of Android Game Programming in CS2. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (Arequipa, Peru) *(ITiCSE '16)*. ACM, New York, NY, USA, 5–10. https://doi.org/10.1145/2899415.2899470

Daniel C. Cliburn and Susan M. Miller. 2008. What Makes a "Good" Game Programming Assignment? *Journal of Computing Sciences in Colleges* 23, 4 (2008), 201–207.

Paul Gestwicki and Fu-Shing Sun. 2008. Teaching Design Patterns Through Computer Game Development. *J. Educ. Resour. Comput.* 8, 1, Article 2 (mar 2008), 22 pages. https://doi.org/10.1145/1348713.1348715

James Hannam. 2024. James Hannam. https://jamesahannam.wixsite.com/portfolio. [Accessed 22-Jan-2024].

Steffan Hooper, Burkhard C. Wünsche, Andrew Luxton-Reilly, Paul Denny, and Tony Haoran Feng. 2024a. Advancing Automated Assessment Tools — Opportunities for Innovations in Upper-level Computing Courses: A Position Paper. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1* (Portland, OR, USA) *(SIGCSE 2024)*. ACM, New York, NY, USA, 519–525. https://doi.org/10.1145/3626252.3630772

Steffan Hooper, Burkhard C. Wünsche, Paul Denny, and Andrew Luxton-Reilly. 2024b. Teaching Game Programming in an Upper-level Computing Course Through the Development of a C++ Framework and Middleware. In *Eurographics 2024 - Education Papers*, Beatriz Sousa Santos and Eike Anderson (Eds.). The Eurographics Association. https://doi.org/10.2312/eged.20241009

IGDA. 2014. IGDA Game Crediting Guide 9.2. Available at https://igda-website.s3.us-east-2.amazonaws.com/wp-content/uploads/2014/08/21170013/IGDA_Game_Crediting_Guide_Draft_9-2-EN-2014.pdf.

IGDA. 2023. IGDA Game Crediting Guide 10.1. Available at https://drive.google.com/file/d/1jhs6vG3Qieu4tjJ6ImqhZ21_pMHkDLxM/view.

IGEA. 2023. Australian Game Development Survey FY 2023 Report. Available at https://igea.net/wp-content/uploads/2023/12/AGDS-2023-Report-Final.pdf.

Dan Irish. 2005. *The Game Producer's Handbook*. Course Tech. Press, BOS, MA, USA.

Randolph M. Jones. 2000. Design and Implementation of Computer Games: A Capstone Course for Undergraduate Computer Science Education. In *Proceedings of the 31st SIGCSE Technical Symposium on Comp. Sci. Education* (Austin, Texas, USA) *(SIGCSE '00)*. ACM, New York, NY, USA, 260–264. https://doi.org/10.1145/330908.331866

Ben Kenwright. 2016. Holistic Game Development Curriculum. In *SIGGRAPH ASIA 2016 Symposium on Education* (Macau) *(SA '16)*. ACM, New York, NY, USA, Article 2, 5 pages. https://doi.org/10.1145/2993352.2993354

Diana Kirk, Andrew Luxton-Reilly, Ewan Tempero, Tyne Crow, Paul Denny, Allan Fowler, Steffan Hooper, Andrew Meads, Asma Shakil, Paramvir Singh, Craig Sutherland, Yi-Chien Vita Tsai, and Burkhard Wuensche. 2024. Educator Experiences of Low Overhead Student Project Risk Management. In *Proceedings of the 26th Australasian Computing Education Conference* (Sydney, NSW, Australia) *(ACE '24)*. ACM, New York, NY, USA, 58–67. https://doi.org/10.1145/3636243.3636250

Amruth N. Kumar and Rajendra K. Raj. 2024. Computer Science Curricula 2023 (CS2023): The Final Report. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2* (Portland, OR, USA) *(SIGCSE 2024)*. ACM, New York, NY, USA, 1867–1868. https://doi.org/10.1145/3626253.3633405

Amruth N. Kumar, Rajendra K. Raj, Sherif G. Aly, Monica D. Anderson, Brett A. Becker, Richard L. Blumenthal, Eric Eaton, Susan L. Epstein, Michael Goldweber, Pankaj Jalote, Douglas Lea, Michael Oudshoorn, Marcelo Pias, Susan Reiser, Christian Servin, Rahul Simha, Titus Winters, and Qiao Xiang. 2024. *Computer Science Curricula 2023*. Association for Computing Machinery, New York, NY, USA.

Luis Levy and Jeannie Novak. 2009. *Game Development Essentials: Game QA & Testing* (1st ed.). Delmar Learning.

Mark C. Lewis and Berna Massingill. 2006. Graphical Game Development in CS2: A Flexible Infrastructure for a Semester Long Project. *SIGCSE Bull.* 38, 1 (mar 2006), 505–509. https://doi.org/10.1145/1124706.1121499

Andrew Luxton-Reilly and Paul Denny. 2009. A Simple Framework for Interactive Games in CS1. *SIGCSE Bull.* 41, 1 (mar 2009), 216–220. https://doi.org/10.1145/1539024.1508947

Andrew Luxton-Reilly, Ewan Tempero, Nalin Arachchilage, Angela Chang, Paul Denny, Allan Fowler, Nasser Giacaman, Igor Kontorovich, Danielle Lottridge, Sathiamoorthy Manoharan, Shyamli Sindhwani, Paramvir Singh, Ulrich Speidel, Sudeep Stephen, Valerio Terragni, Jacqueline Whalley, Burkhard Wuensche, and Xinfeng Ye. 2023. Automated Assessment: Experiences From the Trenches. In *Proc. of the 25th Australasian Computing Education Conference* (Melbourne, VIC, Australia) *(ACE '23)*. ACM, New York, NY, USA, 1–10. https://doi.org/10.1145/3576123.3576124

Stefan Marks and Sebastián Gil Parga. 2023. Computer Graphics and Extended Reality Courses for the Programmophobic. In *SIGGRAPH Asia 2023 Educator's Forum* (Sydney, NSW, Australia) *(SA '23)*. Association for Computing Machinery, New York, NY, USA, Article 3, 8 pages. https://doi.org/10.1145/3610540.3627004

Michael Mateas and Jim Whitehead. 2007. Design Issues for Undergraduate Game-Oriented Degrees. *Proceedings of the 2007 Microsoft Academic Days on Game Development in Computer Science Education* (2007), 85–89.

Mike McShaffry and David Graham. 2012. *Game Coding Complete*. Delmar Learning.

Robin Isfold Munkvold. 2017. Game lab: A Practical Learning Approach for Game Development. (2017).

Ivan Adriyanov Nikolov. 2023. But Worse: Remaking Famous Games on a Budget as a Game Development Course. In *17th International Technology, Education and Development Conference*. IATED, 2991–2997.

NZGDA. 2022. NZ Interactive Media Industry Survey 2022. https://nzgda.com/news/nz-interactive-media-industry-survey-2022/. [Accessed 22-Jan-2024].

Ian Parberry, Max B. Kazemzadeh, and Timothy Roden. 2006. The Art and Science of Game Programming. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education* (Houston, Texas, USA) *(SIGCSE '06)*. ACM, New York, NY, USA, 510–514. https://doi.org/10.1145/1121341.1121500

Ian Parberry, Timothy Roden, and Max B. Kazemzadeh. 2005. Experience with an Industry-Driven Capstone Course on Game Programming. *ACM SIGCSE Bull.* 37, 1 (2005), 91–95. https://doi.org/10.1145/1047124.1047387

Chao Peng. 2015. Introductory Game Development Course: A Mix of Programming and Art. In *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*. 271–276. https://doi.org/10.1109/CSCI.2015.152

Martin Ponichtera. 2024. Martin Ponichtera. https://ponichtera.wixsite.com/portfolio. [Accessed 22-Jan-2024].

Yolanda Rankin, Bruce Gooch, and Amy Gooch. 2007. Interweaving Game Design into Core CS Curriculum. In *Proceedings of the 2007 Microsoft Academic Days Game Development Conference, Nassau, Bahamas, February*, Vol. 21. 2007.

Albert D. Ritzhaupt. 2009. Creating a Game Development Course with Limited Resources: An Evaluation Study. *ACM Trans. Comput. Educ.* 9, 1, Article 3 (mar 2009), 16 pages. https://doi.org/10.1145/1513593.1513596

Ben Stephenson, Mark James, Nigel Brooke, and John Aycock. 2016. An Industrial Partnership Game Development Capstone Course. In *Proceedings of the 17th Annual Conference on Information Technology Education* (BOS, MA, USA) *(SIGITE '16)*. ACM, New York, NY, USA, 136–141. https://doi.org/10.1145/2978192.2978214

Nathan R. Sturtevant, H. James Hoover, Jonathan Schaeffer, Sean Gouglas, Michael H. Bowling, Finnegan Southey, Matthew Bouchard, and Ghassan Zabaneh. 2008. Multidisciplinary Students and Instructors: A Second-Year Games Course. *SIGCSE Bull.* 40, 1 (mar 2008), 383–387. https://doi.org/10.1145/1352322.1352269

Thomas Suselo, Burkhard C. Wünsche, and Andrew Luxton-Reilly. 2017. The Journey to Improve Teaching Computer Graphics: A Systematic Review. In *Proceedings of the 25th International Conference on Computers in Education (ICCE 2017)*. APSCE, Christchurch, New Zealand, 361–366.

Taito. 1978. Space Invaders. [Arcade].

Mark J. Taylor and Michael Baskett. 2009. The Science and Art of Computer Games Development for Undergraduate Students. *Comput. Entertain.* 7, 2, Article 24 (jun 2009), 9 pages. https://doi.org/10.1145/1541895.1541904

Bian Wu and Alf Inge Wang. 2012. A Guideline for Game Development-Based Learning: A Literature Review. *International Journal of Computer Games Technology* 2012, Article 8 (jan 2012), 1 pages. https://doi.org/10.1155/2012/103710

José P. Zagal and John Sharp. 2011. A Survey of Final Project Courses in Game Programs: Considerations for Teaching Capstone. https://dl.digra.org/index.php/dl/article/view/590. In *Proceedings of DiGRA 2011 Conference: Think Design Play*. DiGRA, Tampere.