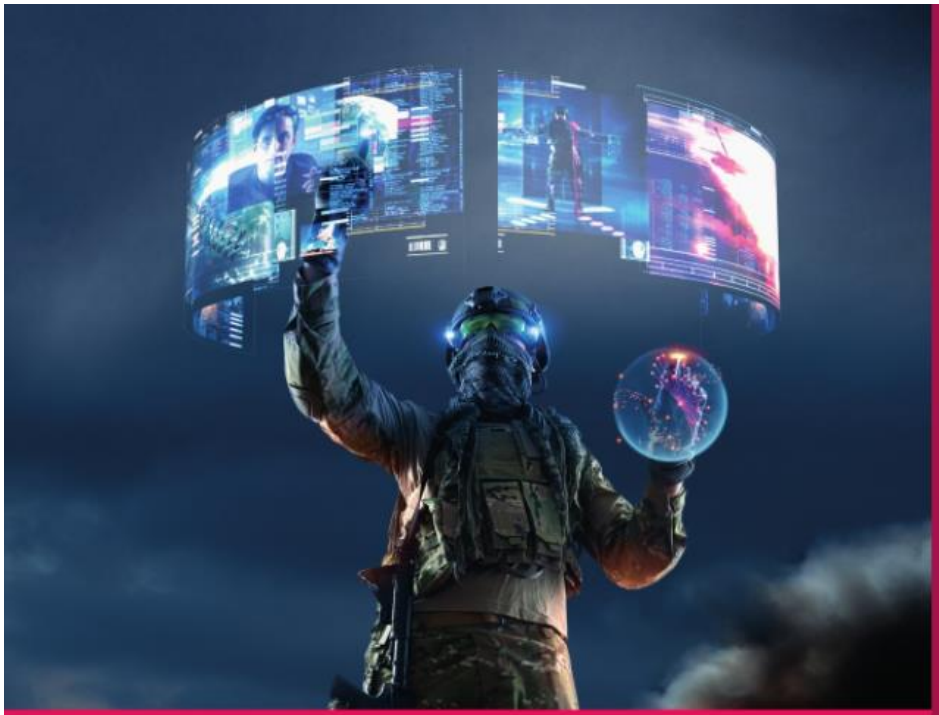

Computer Games Development 8

Getting to know your unit

Assessment
You will be assessed by a series of assignments set by your tutor.

The UK games industry is one of the fastest growing job markets today. The UK is a world leader in producing amazing, interactive experiences for desktop PCs, consoles, and handheld and mobile devices. The computer games industry is a team-based and fast-moving area that involves many different job roles. The production of computer games requires an understanding of the whole process from concept art through to formal testing. You may want to become a games programmer, concept artist or 3D animator but, for whichever role you want to focus on in the future, you will have to a good understanding of the entire games development process first. In this unit, you will investigate how and why people play games, what is changing in games hardware and development, and how to design and create games.

How you will be assessed

This unit will be assessed by a combination of theory, design and practical tasks set by your tutor. Throughout this unit, you will find assessment practice activities that help you work towards your assessment. Completing these activities will not mean that you have achieved a particular grade, but you will have carried out useful research or preparation that will be relevant when it comes to your final assignment.

In order for you to complete the tasks in your assignment successfully, it is important to check that you have met all of the Pass grading criteria. You can do this as you work your way through the assignment.

If you are hoping to gain a Merit or Distinction, you should make sure that you present the information in your assignment in the style that is required by the relevant assessment criteria. For example, Merit criteria require you to analyse and discuss, and Distinction criteria require you to assess and evaluate.

The assignment set by your tutor will consist of a number of tasks designed to meet the criteria in the table. This is likely to consist of a written assignment but may also include activities such as:

- writing an article about the current trends in gaming
- designing a game from a set client brief
- creating and testing a 2D or 3D game level.

Assessment criteria

This table shows what you must do in order to achieve a **Pass**, **Merit** or **Distinction** grade, and where you can find activities to help you.

Pass	Merit	Distinction
Learning aim A Investigate technologies used in computer gaming		
A.P1 Explain social and technological trends of computer games. **Assessment practice 8.1**	**A.M1** Discuss how current and emerging technologies impact on how games are designed and developed to meet the requirements of the users and the larger computer games industry. **Assessment practice 8.1**	**A.D1** Evaluate the impact of current and emerging technologies on the design and development of computer games to meet the requirements of the users and the computer games industry. **Assessment practice 8.1**
A.P2 Explain how current and emerging technologies impact on computer games design and development. **Assessment practice 8.1**		
Learning aim B Design a computer game to meet client requirements		
B.P3 Produce designs for a computer game that meet client requirements. **Assessment practice 8.2**	**B.M2** Justify decisions made, showing how the design will fulfil its purpose and client requirements. **Assessment practice 8.2**	**BC.D2** Evaluate the design and optimised computer game against client requirements. **Assessment practice 8.2**
B.P4 Review the designs with others to identify and inform refinements. **Assessment activity 8.2**		
Learning aim C Develop a computer game to meet client requirements		
C.P5 Produce a computer game to meet client requirements. **Assessment practice 8.2**	**C.M3** Optimise a computer game to meet client requirements. **Assessment practice 8.2**	**BC.D3** Demonstrate individual responsibility, creativity and effective self-management in the design, development and review of a computer game. **Assessment practice 8.2**
C.P6 Test a computer game for functionality, usability, stability and performance. **Assessment practice 8.2**		
C.P7 Review the extent to which the computer game meets client requirements. **Assessment practice 8.2**		

Getting Started

It is important to understand all of the different roles involved in the computer games development workflow. Write down a list of all the different jobs that you think are involved in the making of a computer game, splitting your list into different categories, such as artistic or technical.

A Investigate technologies used in computer gaming

Computer and video games are big business and a continuing source of exciting and creative jobs that require strong IT skills and original ideas. The next big successful title could be just around the corner and one of the most exciting elements of the computer games industry is how small teams with great ideas can become international success stories. Different **genres** can grow out of new titles and create entire new fan bases of dedicated gamers, sometimes forming strong communities of players. Many of the recent game genres have grown out of innovative technologies such as virtual reality, augmented reality, new operating systems and advances in streaming.

Gaming trends and society

The original gaming device was a cathode ray tube amusement device in 1947. There have been nearly seven decades of gaming developments since then and people access and enjoy computer games in many different ways. These different trends have led to an ever-evolving industry that seeks to bring interactive entertainment to all areas of society, whether they are casual gamers wanting a few minutes of distraction with Angry Birds™ or hard-core gamers who eagerly anticipate the newest release of the Halo® series.

Figure 8.1: A scene from a video game

Popular genres

Innovation and originality are keys to the success of a game publisher and players are always willing to explore different genres of games. One of the most popular genres in gaming is the First-Person Shooter (FPS) genre where the player sees through the eyes of a character and must fight through different levels to achieve a particular goal, for example Call of Duty®. Genres can grow and change, they can spawn sub-genres that take the original definition of a genre and spin it off in a different direction, or they may become hybrid genres that take elements of other genres and merge them together. An example of this is an online FPS, such as Star Wars™ Battlefront™, which features many players fighting at the same time. This game has all the features of an FPS game but it also fits into the **massively multiplayer online (MMO) game** genre. So it could be called an MMOFPS. To muddy the waters further, this game also allows you to switch to third-person mode so that you can see the character you are playing.

Key terms

Genre - a genre is a category of computer game that describes the style of play, types of challenges and the perspective of the player.

Massively multiplayer online (MMO) game – a game played by multiple players, across the internet, all online at the same time.

Role-playing games (RPGs) have been popular since before games were created digitally. Classic pen and paper games such as Dungeons and Dragons™ saw players work against a dungeon master, having chosen a particular character and, as the game progressed, their character would grow and improve as they made choices. This role-playing concept works very well in computer game format and some RPGs, such as The Elder Scrolls V: SKYRIM®, have had massive success. Other games developers have noticed the appeal of RPG game players improving their characters and have been adding this feature to titles from other genres, such as Call of Duty®, enabling them to 'level up' and receive perks and rewards.

The nature of constant change within computer games, of borrowing of ideas from other genres and adapting to player feedback, means that it is impossible to write a definitive list of genres.

Figure 8.2: Games bring people together as a common interest and popular pastime

Types of player

As computer game titles can be categorised into different genres, it is also important to consider different types of player. When a game studio pitches an idea for a new title to a publisher, one of the most important factors they have to consider is the audience of the game they want to make. It is impossible to make a game that will please everyone as players have different tastes and interests, and psychologists such as Richard Bartle and David Keirsey have even made academic studies into how and why people play games. The table below (Table 8.1) looks at the main factors that determine the types of player.

The table above represents basic demographics, that is, measurements used to put people into different categories so that their likes and dislikes can be understood more easily. However, there are other considerations to look at when designing games.

In recent years, as home broadband speeds have increased, online games have soared in popularity and the previous generation of home consoles (PlayStation 3™, Xbox 360®) were designed to appeal to people who wanted to play together in engaging, online worlds. Our most recent generation of consoles (Wii U™, PlayStation 4™ and Xbox One™) have all included the ability to play online and stream content so that other people can watch the play and comment on the skills of the players.

Research

How do people play games together? Try to list as many different ways in which traditional games (pen and paper, board games etc) and computer games provide multiplayer activities.

Key terms

Casual gamers – people who only play games for short periods of time and prefer simpler games.

Immersive – a term which refers to how focused you are on the experience that you are having. An immersive game will keep your attention for long periods of time and block out distractions. It should make you enjoy the game more, but only if you have the time to spend on it.

Franchise – a series of game titles that feature the same world, the same characters or the same setting.

Table 8.1: Main factors that determine the different types of computer game players

Age range	In the UK, games are rated by PEGI. PEGI decide what is appropriate for different age groups. Game designers will tend to create bright, cartoon adventures for younger players and darker, more realistic worlds for adult players.
Gender	Despite being commonly considered to be a 'boy thing', gamers are split between male and female, with roughly 60 per cent of players being male and 40 per cent female. Game designers often design children's games for a specific gender, but the older the target audience gets, the less targeted to a particular gender the content becomes.
Time commitment	Possibly one of the most important factors is how much time a person is willing to spend playing a game. This can affect the design of a game greatly. **Casual gamers** will happily load a game for a few minutes at a time. They may play on mobile devices while travelling or download digital games that cost a lot less than boxed games. Casual gamers will not be particularly loyal to one genre, worried about saved games or necessarily interested in sequels.
	Players who want a much more **immersive** experience will spend hours exploring their favourite worlds. They will be loyal to particular brands and spend a lot of their spare time playing games. Immersive gamers may also be referred to as 'hard-core gamers'. Some people may criticise the time they spend on games, but is it any different from spending hours watching TV?
Theme choice	The content or style of a game and any ideas that tie all of its features together are called its theme. People may choose a game based on its story, setting or design. There are many different game themes: a fantasy adventure, a realistic war game, a puzzle game with fun characters or a game that follows the stories and characters from a well-known film or book **franchise**.

Game production

The production of a game is a complicated event. It usually begins with an idea being pitched by a **game development studio** to a **game publisher**. This pitch will contain an overall concept of the game with details about the characters, **game mechanics** and who the game is being designed for, that is, the audience. Without an identified audience, there would be no financial incentive for making the game. Sometimes the publisher will approach the studio with an identified audience or game genre and/or theme and ask them to create a game for that market. After the studio has been given the funding to start development, they will begin by creating **concept art** that illustrates the graphical style and theme of the game (see Figure 8.3).

Key terms

Game development studio – a team of people who create computer games.

Game publisher – a company that releases games to shops or online platforms, and pays for development.

Game mechanics – the way a game world works, its features and its rules such as double jumps or collecting items.

Concept art – drawings and paintings created before a game is developed to show how the game world should look and feel.

Figure 8.3: Concept art by Ewon Harding

The concept art will be referenced throughout the game's development so that everyone on the development team knows what the game should look like, and the kind of feelings that the game should be evoking in the player.

Mainstream publishers

Some publishers are large international companies, known as mainstream publishers, such as Electronic Arts (EA™ Games) or Nintendo®. These mainstream publishers are responsible for funding the production of many different game titles at the same time. Smaller publishers, such as Telltale Games®, will usually specialise in one particular genre and will often release only one game at a time.

Indie games

When development of a game is started before it has a publisher, it is known as an indie game. Studios who make indie games fund the development themselves. This means that they do not have the same restraints placed on them by publishers during development so they can make their own decisions about the content and style of the game. This often leads to innovative and original game designs but it also means that the studios can struggle to pay the bills, as they do not make any money until the game has a publisher. When a studio has proved that there is a market for their game, through research or testing, a publisher will fund their project. Good examples of successful indie games are Minecraft™, Super Meat Boy™ and No Man's Sky™.

Indie games are able to start making a profit when they find a publisher. Often the profit is fed back into the development of their next game, before it gets a publisher. Indie games usually get published as digital downloads because getting a game disc printed, boxed and shipped to game shops all around the world costs a lot of money. It is partly because there are so many good indie games now that digital download providers (such as Steam™, PlayStation™ Store) have become so popular. Mainstream publishers will now decide before development if a game is going to be printed on disk or released through a digital platform. If they expect to sell to a wide audience then they will make the investment in a boxed game sold in a shop, but if the game is more niche then it may only be released digitally, as there is a greater financial risk in printing on disc because it costs more.

Crowdfunding

Indie developers are able to raise funds directly from players through crowdfunding websites such as Kickstarter™ or Indiegogo™.

Crowdfunding is a recent phenomenon and it has really exploded over the last ten years to a point where it has gone from being a gimmick, which could be used to top up the finances of a struggling studio, to a viable means of funding large-scale projects. Virtual reality hardware such as the Oculus Rift™ would not exist without crowdfunding. Equally, some very successful games have been funded through this model, for example Broken Age™, which was one of the first games to raise far more funding than it asked for. Star Citizen™ is the most successful crowdfunded game to date, having made over £25 million.

The crowdfunding model differs from traditional game development not only in how the money is raised but also in the inclusion of 'stretch goals'. This means that developers will commit to a certain scope for the game, but given additional funding they would make additional promises. These 'stretch goals' may be different platforms for release, extra levels or a virtual reality mode. One of the disadvantages of crowdfunded games is the delays to titles, over which customers have no control, as extra features can push back delivery dates and people can often wait a long time for their games.

Free-to-play

Another model of game production is the free-to-play model, which has become a mainstay of mobile apps games development. Using this approach, smaller studios and publishers release games for free and then include a number of paid-for upgrades or features. The idea is that players will get hooked on the game when they can play it for free and will then be willing to spend money unlocking extras. These free-to-play games are also known as freemium titles. Some players find it frustrating to begin a game that they think is free only to find later that they have to pay to keep playing. Publishers of free-to-play games can avoid making players pay for the games after a certain period by placing adverts in the games, which pay for the cost of development.

Link

The development of mobile games apps is covered in *Unit 7: Mobile Apps Development*.

Artificial intelligence

Artificial intelligence (AI) is the name given to the programming that makes machines (or non-playable characters (NPCs), in the context of games) seem like they are thinking for themselves.

Programmers have to think about how an NPC in a game should react to where they are or what they are supposed to be doing. A soldier guarding a gate should be looking in certain directions but should have moments where their attention lapses so that the player can sneak past. All of this character behaviour has to be coded into the game.

Modern game AI has developed to the point where enemies are able to have realistic reactions to players: such as an enemy blocking an attack in a fighting game, or more strategic behaviour such as enemies responding to patterns in the player's choices by avoiding a 'duck and cover' attack or trying to flank enemies, in multiplayer games. The more sophisticated the AI, the more complicated the programming required to create it.

Search algorithms

Different **algorithms** are used when designing AI and programmers often use search algorithms, such as the A* Algorithm, which is a path-finding algorithm. A search algorithm allows the AI attached to a NPC to solve problems, such as the problem of where to go next. The problem of deciding the best route to take from one point to another is quite easy for humans: we look at the area around us and decide the best way to get somewhere using the smallest amount of effort. It is not as easy for an NPC (often an enemy) in a game; the NPC has no real concept of where it is or where it should be. Therefore, when a player is seen by the enemy at the end of a hallway that contains several crates, barrels and piles of rubble, it needs an algorithm to figure out how to get down the hallway to the player. The A* Algorithm would break the hallway and obstacles down into a series of steps and work out the best set of steps to get the NPC from its starting point to its goal. There would be a choice of different routes to take and the algorithm uses mathematical optimisation to work out the best route to take.

Key term

Algorithm – a set of instructions that are executed in order to solve different computer problems.

Mathematical optimisation

Mathematical optimisation is the selection of the best solution to a problem when given a selection to choose from. It forms a part of all AI and is often tied to a game's difficulty setting. Most game NPCs would have the ability to get rid of a player quickly, if programmed correctly, but that would render the game impossible. Therefore, a game's difficulty setting will change the possible solutions available to NPCs, from a simple approach such as lowering the enemy's health value or making a less powerful adversary, to more sophisticated AI such as the enemy being able to change tactics.

Logic

An important factor in creating realistic AI is considering the logic and patterns of behaviour that a human would normally follow. Does the enemy behave in a way that is logical and reasonable? If it emerges in the testing phase of a game's development that the answer is no, then the AI programmers must go back to their code and improve it until a player is not puzzled by the behaviour of an enemy. This is a difficult task as games are very complex, interactive software applications and it is hard to predict what players might do in a game. In fact, players often cheat by finding an 'exploit' or vulnerability in a game's AI. This often leads to developers having to find a fix that they then patch onto the game as a download after the game has been released.

Emerging technology

One of the things that makes the computer games industry so exciting is the rapid pace of development and new ideas that change our game playing experiences. New ideas are driven by improvements and innovations in the hardware and software that we use to play games.

Research

What improvements to games hardware have been made in the past few years? Have a look at the technical specifications and features listed by different hardware manufacturers and see if you can come up with a list of the technological features that define the current generation of games platforms.

Virtual reality

One of the fastest growing areas is virtual reality (VR). Back in the 1990s, VR was set to be the next big thing but the headsets were very expensive and the processing power was too slow, so the technology never took off. However, more recently crowdfunding has brought about the first version of the Oculus Rift™ headset, which has proved to be an incredible success and spawned imitators from Valve™ and Sony®.

Figure 8.4: Virtual reality is creating a whole new way of engaging with games

VR in its current form consists of two screens, one over each eye, with special lenses that magnify the screens so that they fill your field of vision (see Figure 8.4) . Added to the headsets are accelerometers, which measure which direction you are looking in, and gyroscopes, that measure how much you have turned your head. Often a positional tracker is paired with the headset, which follows your head's position in space so that you can move forwards and backwards. This positional tracker can be combined with motion controllers, and even a treadmill, to create total movement in a virtual world.

Augmented reality and wearable technology

Microsoft® have taken a different approach to the new headset phenomenon and have focused on augmented reality (AR) instead of VR. AR uses cameras to capture the real world and layers virtual game assets over the top, so that they appear to exist in the real world (see Figure 8.5).

AR already existed on various mobile apps and in game form on the Nintendo® 3DS™ and PlayStation Vita™ but Microsoft®'s HoloLens™ seeks to outdo them all by creating engaging experiences with virtual characters in the real world.

Once you have exhausted all the games on your PC, console, mobile and handheld games devices, you can look to smart watches to top up your gaming urges (see Figure 8.6). The Apple® and Android™ smart watch app stores both have a growing range of simple games, which you can play on your wrist. These games are restricted by the size of the screen that they are played on, but sometimes a physical restriction can be the catalyst for an amazing game idea. Lifeline™ is an example of an Apple® Watch® game, where the player receives messages from an astronaut stranded on an alien moon. The game's interface is simple text but it is an engaging and exciting story with delays built in so that players are not staring at their watches for hours on end.

Figure 8.5: Whilst virtual reality places you in a different world, augmented reality brings virtual objects into our world

Figure 8.6: The Apple® Watch gives you access to apps on your wrist

Digital distribution

It is not just headsets that are changing the way we play games. Digital distribution platforms such as Steam™, GOG™ and PlayStation™ Store have changed how we buy games. One of the most dominant games providers, Steam™ by Valve™ is also one of the best places to get indie games, which can often cost a lot less than console disc-based titles. Disc-based games are often priced the same as their digital equivalents on the console's own digital stores, as there are not any incentives for console publishers to discourage their audience from buying games in a traditional shop, because if the shops cease to exist then there are fewer places to buy the console.

Steam™ has created its own operating system (OS). Steam OS is a free, **Linux®**-based OS that prioritises gameplay and is available on 'living room' PCs. 'Living room' PCs are compact PCs that are connected to a TV and are designed to be used in a similar way to a traditional games console, by using the digital platform as the main source of content.

Key term

Linux® – an operating system (OS) which is released 'Open Source', meaning that the source code that creates it can be downloaded and adapted by anyone. Due to its adaptability, it is very popular and comes in lots of different versions such as Steam™ OS.

Streaming

Streaming has become a huge part of games culture with websites such as Twitch™ and YouTube™ making gameplay videos highly popular because they enable people with limited budgets the opportunity to preview a game that they might be interested in buying. Many people are used to streaming music and films to different devices, and games can be streamed too, using services such as PlayStation Now™. Subscribers can rent games for set periods and download them to their device, be it a console, PC or a smart TV.

Discussion

Which of these emerging technologies would engage you as a gamer? Discuss, in a group, who you think these emerging technologies would engage and whether or not new audiences for games may grow out of these emerging technologies.

Security of integrated services and multiplayer environments

As game systems become more reliant on digital services and streaming content, there is an increased requirement for users to share their personal data with the services that they are using. Most of the digital platforms require stored credit card details and they will often ask customers to connect their social media sites so that customers can share details of what games achievements they have gained or levels they have completed. This presents a big security risk, as the information could be used by criminals to steal a customer's identity. For example, Sony® was the victim of a series of cyber attacks in 2014. All the companies offering games platforms have to be very careful that new updates and upgrades to their software do not inadvertently create a breach that criminals can exploit.

MMO game providers can also be the targets of hackers and they often have less security in place than an online bank or shop. It is important that they protect their user's data because, despite being a game rather than a bank, they are still holding the same level of sensitive personal data about customers.

Players do not just have to worry about the security systems in place on MMO games, but they also have to be careful about the information that they give out if they are having conversations with strangers in the game through text or voice chat. Criminals are able to get a lot of information out of players without them realising it, by pretending to be friendly. Before they know it, the player's password has been guessed and they are locked out of their own account. (This could be their login for an MMO such as World of Warcraft® or a digital distribution account such as Steam™ or Google Play™.)

Gaming technology

Keeping up to date with games technology, and how it is changing, is one of the main challenges for anyone involved in the computer games industry.

Benefits and limitations of different platforms

Players face the added challenge of choosing which platform to invest in. While developers may often want to create games for the most popular platforms, each one comes with its own benefits and limitations.

Hardware

Most gaming hardware is made of the same core components to allow digital information to be displayed on screen and interacted with by the player.

Central processing unit

The central processing unit (CPU) is the brains of the computer. It carries out all of the instructions sent to it by a computer program using thousands of tiny switches to perform arithmetic, logic and input/output. In the case of a game program, the CPU is responsible for working out all of the game's system requirements. Most modern PCs use an Intel® or AMD CPU and the performance of PC games (such as that of Steam™ titles) is dependent on the power of the processor. For example, The Witcher® 3 is a a huge open world RPG with very good graphics and its minimum system requirements include an Intel® Core™ i5 processor that runs at a speed of 3.3GHz (a fast processor which is not sold cheaply), but its recommended requirement is an even more expensive Intel® i7 processor running at 3.4Ghz. Given this, many PC gamers will overclock their

Table 8.2: Benefits and limitations of different platforms (then update subsequent table numbers)

Platform	Type	Benefits	Limitations
Windows® PC	Personal computer or laptop (desktop)	Easy games development; broad user base; access to Steam™	No standardised technical specifications; some games will not run on lower spec PCs
Mac®	Personal computer or laptop (desktop)	Works with most digital platforms; powerful systems with fast CPUs	Fewer games are released for Mac®
PlayStation 4™ and Xbox One™	Console	Hugely popular consoles; strong hardware; OSs dedicated to playing games	Can be expensive to develop for; getting disc distribution requires working with large publishers
Nintendo® Wii U™	Console	Large fan base; not too many titles on the shelves	Very specific tablet-based controls mean that games cannot be ported onto other systems
Apple® iOS devices (phones and tablets)	Mobile devices	Hugely popular with owners willing to invest in higher priced games than non-iOS® mobile apps	All products have to be approved by Apple® who can have quite strict standards; development usually restricted to Mac® unless using a games engine
Android™ devices (phones, tablets and notebooks)	Mobile devices	Much cheaper development costs than Apple®; open source system	So many different types of devices means quality testing can be difficult
Adobe® Flash®	Web-based platforms	Easy to animate and create great visuals on	Flash® does not have simulated physics; due to security concerns, many browsers are discontinuing their Flash® support
HTML5	Web-based platforms	Supported on all web browsers, desktops and mobile devices; no plugins necessary	No built-in support for 3D, gamepad or to save games

CPUs. 'Overclocking' means to force the CPU to run faster than its recommended manufacturer speed and, while it can be done with stability by changing the voltage, this will invalidate the warranty and put your PC at risk.

Games consoles are able to use slower CPUs because they optimise their games to run with fewer details. For instance, they may use less real-time shadows or lower-resolution textures. They also do not have to worry about running large operating systems in the background, like Windows® or OS X®.

Mobile devices, such as smart phones, run with much slower CPUs, which is why they are unable to run large-scale games with complex graphics.

Link

For more information about CPUs, see *Unit 4: Programming*.

Graphics processing unit

A graphics processing unit (GPU) works alongside the CPU and its sole responsibility is to manage the production of images on the display. The more complex the graphics, the more work for the GPU. When playing PC games, the GPU can be part of the main **motherboard** but is more commonly a separate card connected to the motherboard through an expansion slot. Most GPUs have their own separate random access memory (RAM) (see below) and PC games will often specify the minimum power of the GPU needed by a game to run. Consoles also have their own GPUs, as do mobile devices.

Key term

Motherboard – a circuit board within a PC that connects all the main components.

Memory and storage

Memory is split into two different types: read-only memory (ROM) and random access memory (RAM). ROM is used in games as storage for the game's software. This is usually a DVD or Blu-Ray disc, (downloaded games from Steam or mobile app stores are stored on a hard drive or flash memory) but cartridges were used in the past in consoles and handheld devices. RAM storage is used to temporarily store data when the processor is working on it, and the more RAM available, the faster things usually

run. Consoles and mobile devices use the amount of RAM they have as a selling point of their systems whereas on a PC it is a lot cheaper and easier to upgrade the amount of available RAM.

Long-term writable storage, unlike ROM that is read only, usually comes in the form of a hard drive or flash memory that will store not only the game data itself but also user-based data such as game saves and leaderboards. As games get more popular and people want to change devices when new models come out, cloud storage for game data has become a staple feature of consoles. Letting players store game data and game saves in the cloud means that they are able to continue playing from the same point if they upgrade or change their machine.

Output

Sound and display graphics are major elements of games and, as such, form the biggest share of the output. Gaming PCs and consoles use TVs or monitors to display the graphics but some, like the Wii U™, will have secondary screens for displaying additional information or providing a screen to use when someone else wants to watch TV. Smartphones and tablets are currently in a hardware war to outdo each other's display capabilities, with 1080 pixel screens becoming the standard and 4K screens providing unbelievable detail on a device that you can carry around in your pocket.

Sound is important, as it is one of the main ways in which the game communicates with the player without distracting them with text. A positive sound effect will inform the player that something good has happened and part of a game's learning curve is to remember which sound effects mean which benefits or punishments.

Another output of a game could be haptic feedback, which is a physical sensation such as a tremble or vibration, which is usually provided by controllers (see Figure 8.7).

Figure 8.7: Game controllers can provide a rumble feature which provides extra feedback about events in game

Input

The way in which a player inputs information into a game is more crucial than any other aspect of software for computer games. The slightest delay in the interpretation of input by a game could mean the difference between life and death for your character.

Many PC players will argue that keyboard and mouse controls are better than those of a gamepad controller, but it really comes down to personal preference. Keyboard and mouse controls offer more customisability but players can use gamepad controllers when sitting on a sofa rather than at a desk.

Recent developments in voice and kinetic controls have created new and innovative ways to control games that not only challenge designers to come up with new ways to input instructions into games, but also open gaming up to people with physical disabilities who may struggle to hold traditional gamepads.

Touch input has become a big consideration as games for mobile devices have become more commonplace, because users will need to use the touchpad of their smartphone or tablet to play games. Some gamepads now include touchpads so that casual gamers are tempted to move across to console titles.

Connectivity

As we move into a time where society is always online, gaming devices are required to stay connected to the internet as much as possible. This allows operating systems to perform fixes and updates and for games to download patches to deal with bugs and exploits. PCs and consoles maintain their connections through local area networks (LAN) or wireless connections, while mobile devices can also enjoy wireless through mobile data network connections.

Link

For more on new hardware technologies, see the Emerging technologies section.

Software

Games are software applications, written using a programming language, and run on an operating system. They are a lot more complicated than most software applications but they are written using many of the same programming processes and techniques.

Link

For more information about programming processes and techniques, see *Unit 4: Programming*.

Operating systems

PC games are run on either a Windows®, Mac® OS X® or Linux® operating system. These operating systems differ in popularity and this is the main deciding factor in how many games are made for these systems. While most family homes will have run a Windows® PC, Linux® is a more specialist operating system and Apple® Mac®s are traditionally used by designers. If the priority use of the operating system is not gaming, then fewer games will be made using that operating system. However, digital platforms have started to change this.

PC operating systems are made to work on a number of different hardware configurations (on various different PCs, laptops and notebooks), unlike console operating systems which are designed with far fewer features and for one specific piece of hardware (one version of a console, eg Xbox One™). Console operating systems are known as proprietary systems, which means that the hardware manufacturers create them, and the source code is kept private and is not intended for use anywhere else. Most mobile devices run on either Android™ or Apple® iOS, but some run on Windows®. A newer operating system is Steam OS by Valve™ which has been designed for PCs that are primarily being used for gaming.

Programming languages and graphics options

The programming language in which the game is written can also vary quite a lot depending on the platform that it is intended for. The dominant language for developing games is C++, which is an **object-oriented language** that has been popular for nearly thirty years. Other languages that use a similar structure are C# and Java. C# is used within games engines and Java is used to create Android™ games. Different programming languages tend to be tied to different platforms. Some languages are described as 'light weight' and will only run within a games engine, such as JavaScript. These 'light weight' scripting languages rely on existing assets and features to keep the amount of code low and to produce games quickly, but they may run slower as a result.

Key term

Object-oriented language – uses code that is organised into objects, which can be used to make it run in a fast and robust manner.

Graphics software also needs to be considered. Systems use application programming interfaces (APIs) for managing the tasks related to the software and the GPU. The APIs are largely sets of routines and protocols for making the development of games easier. DirectX® is a Microsoft® graphics API and OpenGL® is the open source alternative.

Device drivers

Device drivers are another important type of software used for running games. The drivers are responsible for identifying the hardware devices that have been connected to the operating systems. Drivers tell the PC, console or mobile devices which hardware is trying to communicate with it and how it should work. This is how peripheral devices connect to games, such as newer controllers, microphones or even dance mats. Console manufacturers such as Nintendo® or Sony® will often release extra hardware devices in order to prolong the lifespan of the system and keep players interested.

Audio options

Audio is another important area of software. The music and sound effects are a massive part of the feeling and ambience that designers want to create for a game and one of the best ways to evoke feeling in a player is through music. A game will have different music in the different levels and these will be similarly themed to generate a common link across the game. Sound effects are used to teach players when they have done something positive or negative. Music files can be quite large and developers have created a number of different file formats in order to make the games run more smoothly, especially for online games. These file types are WAV, MP3, FLAC and AAC.

PAUSE POINT

Create a technical specification for both a photorealistic 3D, online FPS game and one for a 2D casual web game. What are the main differences and what decisions about software do you have to make?

Hint Think about the components that would be required: the basic requirements are a CPU, GPU and memory, but what extra hardware would make the gaming experience more complete?

Extend Can you find examples of games that stretch the capabilities of their platforms? Are there any 3D games for the web? If so, what sacrifices do they seem to make in order to run smoothly?

Games engines

Writing a computer game from scratch is a large undertaking and can involve writing thousands of lines of code. One way to make this easier is to use a games engine. A games engine is a piece of software, which is designed to make games. Most games engines are a combination of designer and programming environments in which the user is able to place all of the graphical assets for a level and then write the code that makes it interactive. There are a number of popular games engines in use today. Some of these are proprietary, meaning that they are only available to particular publishers or the large games studios that own them. However, some others are available to anyone, either for free or for a subscription fee. Popular games engines for multiple platforms are Unreal® Engine 4 or Unity® (see Figure 8.8), which are able to build games for PC, consoles, the web or mobile devices. Other engines are designed for one particular platform, for example XCode® is designed for Apple®'s iOS.

Figure 8.8: The Unity® Game Engine is a powerful application used to build games quickly and efficiently

Rendering engines

One of the main jobs of a game's engine is **rendering.** This is usually done through one or more virtual cameras that point at the actions within the game that the players need to be focused on. The speed at which the game is rendered is measured in frames per second (FPS). The faster the FPS, the smoother the game appears to the player. When a gaming system is not powerful enough to run the game that the player has loaded, the FPS is usually the first casualty and a game may appear to run slowly or lag.

Physics engines

As all computer game worlds are virtual, all of the events and rules that occur within them have to be programmed. Some of the hardest elements to create involve the laws of **physics** that need to be applied to the world. Most games engines include built-in physics that takes the responsibility away from the programmers and allows for faster game production. Other engines that do not include built-in physics can use APIs to add physics. An example of this is the Box2D physics engine that was used to create Angry Birds™, one of the most popular physics-based puzzle games ever made. Another physics engine is the Havoc® physics engine, which has been used in many popular 3D games such as Assassin's Creed®.

Key terms

Rendering – the process of converting game assets and environments into 2D images that can be displayed on a screen.

Physics – in the real world, gravity, mass and other laws of physics apply naturally but in a world created by a computer programmer the laws of physics have to be made to apply through coding the game correctly.

Collision detection – the process of checking to see which game objects have collided with each other.

Collision detection

Once the objects in a game world have physics, they have the ability to collide with one another. Working out the correct way for objects to collide which each other, and what occurs afterwards, is known as **collision detection** and is another important role of a games engine. Objects in a game world may have solid colliders surrounding them so that game characters are not able walk through them. These colliders may be the sides of a building, the floor or even invisible walls to stop a player falling off the edge of the game world. The player's character will also be surrounded by a collider that detects where their boundary is and stops them from moving through walls. A game character, whether it is 2D or 3D, has a very complex shape and it would be too taxing on a system to make every part of their body a collider. Instead, a capsule-shaped collider is often placed around them (see Figure 8.9) which is why, if you are careful, you can sometimes see a character's arm or leg moving through a solid wall in a game. The more you learn about how games are made the easier it is to spot things like this when you are playing them.

Scripting

A games engine will always provide the ability to add code in order to make things happen in the virtual world that has been created. Code is added using a **scripting language** such as JavaScript, or a full programming language like C++. The Unreal® Engine contains a visual scripting language called Blueprints. This is an easy way to create interaction without having to write lines of code.

Figure 8.9: Character surrounded by capsule-shaped collider in Unreal® Engine to enable collision detection

Graphical nodes are used to create movement, interaction and game mechanics like health or ammunition. Blueprints is similar to Scratch, a visual scripting language used to teach programming, but Blueprints provides a lot more functionality. The downside of using a visual scripting language is that the more complex the game's mechanics, the larger the amount of nodes on screen, and it can soon look like a scramble of lines and nodes. If this happens, it would be better to switch to C++ and write the game in traditional code.

Key terms

Scripting language – a programming language that requires a separate application to run, such as a games engine or web browser.

Graphical nodes – colourful blocks used to represent the different functions of a programming language used to add easy interaction.

Animation

The final responsibility of the games engine is animation. Animation involves making different elements in the game move. There are two types of animation: vertex-based animation (which moves parts of a 3D mesh or 2D sprite) and skeletal animation (which uses a rigged character and sets of different animation cycles). An example of vertex-based animation would be a door opening when the player approaches it or a floating coin spinning around.

The image below (Figure 8.10) illustrates skeletal animation in the Persona system within Unreal® Engine. A character created in 3D can be rigged with a set of virtual bones that are, in turn, attached to different parts of the mesh. When a virtual bone is moved or rotated, the mesh connected to it will also move. The player will never see the bones but the engine knows that they are there and understands how they are supposed to act. The 3D character modeller will rig the character in a design software application such as Autodesk® 3D Studio Max, weight the bones so that the mesh is connected properly and then create different animation cycles such as a walk cycle, run cycle, jump cycle etc. These cycles are a single animation that ends at the same position as it starts so that it can be repeated for as long as the game requires the character to do that action. For example, when a player is pushing forward on the gamepad, the game will repeat the run cycle for as long as the player pushes forward. When a character is imported into the games engine, the animation cycles will be imported too.

Figure 8.10: Skeletal animation in the Persona system within Unreal® Engine

PAUSE POINT Games development can be an expensive process but much of the software mentioned so far is free for learners to use or does not cost anything until a game is published and has made a certain amount of money. Can you find out what the costs are for the software discussed so far?

Hint The websites for each of the different software are the best places to start.

Extend We have discussed game engines, 3D modelling software and operating systems, but graphic design software plays a large part in games development too. What additional software might you need?

Assessment practice 8.1

A.P1 A.P2 A.M1 A.D1

A mainstream games publisher is interested in creating a new game franchise that will have different titles that embrace the latest developments in current games technology. They want to show that they understand the needs of gamers and the different devices on which they can play. They also want to be seen as being part of the next phase in gaming. They have asked you to make a presentation to one of their development studios to help them understand the opportunities that are available.

They want a presentation which:

- explains the current social trends of computer games, including:
  - Different types of players.
  - Who buys what games?
  - How have habits have changed over time.
- explains the technological trends of computer games.
  - What emerging technologies are being developed in the computer games industry?
  - How are existing technologies being used alongside new ones?
- discusses how current and emerging technologies impact on how games are designed and developed to meet the requirements of the users and the larger computer games industry.
  - How is this changing the games that people buy?
  - How have game designs changed to meet them?
  - Who is investing in this technology and what will they gain?
- evaluates the impact of current and emerging technologies on the design and development of computer games to meet the requirements of the users and the computer games industry.
  - What technologies have failed in the past?
  - Are people playing games differently?

Plan

- What am I being asked to do?
- What information do the designers need?

Do

- Am I getting the most up to date information?
- Can I look at how past technology has impacted on game design?

Review

- I can explain what parts of researching the presentation were the hardest.
- I realise where there are still areas of the industry where I have knowledge gaps.

B Design a computer game to meet client requirements

Designing a computer game is a complex process involving team members with many different specialist skills. The design phase is vitally important and rushing this stage can lead to many problems for the development team, or even the players further down the line. However, game development studios are not always in charge of their own deadlines, for example if they are working on a commission for a large publisher. Often they will be given the contract on the understanding that they meet very specific deadlines, so an efficient but also effective design process is crucial.

Computer games design processes and techniques

Understanding the formal methods used to design a game is very important. Each member of the team is trained in their own specialism (art, asset creation or programming) but they all have to understand all the steps in the game design workflow. The first stage of the game design process is the creation of a high-concept design document that will outline all of the game's unique features, storyline, characters and mechanics. This document is then expanded into a game design document (GDD), which covers all of the detailed design specifications for the entire game including how it will be made, how many levels there will be, how the game world works etc. The GDD is then shared between every member of the team and extra documents are written, for example a style guide and a technical specification. These two documents contain even more detailed designs for the specialist teams who will be working on the art or the programming sides of the game.

Mathematical techniques and processes

Maths and logic are a core element of programming and as such they form a crucial part of games design. The technical specification outlines all of the code that is required for the game, the platforms and languages that are being used, as well as details of the developers who will be responsible for creating them. The processes required for creating the functionality of the game and the maths that will make it happen must be considered thoroughly.

Calculations

Calculations are present in all software. Every time we click a mouse button or a gamepad trigger there are lighting fast calculations being performed in order to figure out what should happen as a result. Games use maths in a variety of different ways. The mathematics of graphics is one of the most common uses of this as geometry (the area of mathematics that deals with shape, size and space) is used to position 2D and 3D assets in a virtual world and then manage how they move, rotate and interact with one another. Geometry deals with points, lines, planes and solids and this transfers directly to the 3D modelling of objects where we create vertices, edges, faces and meshes. Games will use geometric calculations as part of their game mechanics. So, if we are playing an FPS where a player must fire a bullet at a target, geometry is used to calculate what happens if the player shoots at a particular angle and from a particular distance. Where would the bullet hit the target if fired in that direction, at that angle and from that distance? Geometry is used to establish which shots would achieve the desired result and which shots would miss. In geometry, the point where a line cuts through a plane is called the intersection, and it would be this value that decides whether the player has been successful or not.

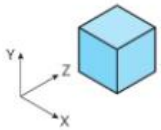

Figure 8.11: All 3D game objects have a position in 3D space which is measured on an XYZ co-ordinate

2D and 3D space

The various different points in game space that need to be monitored, such as the location of a power-up or the exit to the level, will be stored as a vector. A vector is a mathematical way of representing the point in terms of its location on the *x*, *y* and *z* coordinate axes. Based on its distance from the game world's centre point (0,0,0), you can consider the vector value to be the object's address in the game world. Vectors are written as (*x*,*y*,*z*), always in that order, so that all developers and functions within the game understand the vector's location.

Vectors have two main abilities: they can move (or transform) from one place to another or they can be rotated into a new position. The way in which a vector rotates is decided by its pivot point. A spinning coin may have a central pivot whereas a door would pivot from one side. Vector values are also used to describe a direction that something is going to move in, so, if a platform was going to move to (0,0,10), it would move 10 units in the *z* direction but stay in the same position on the *x*- and *y*-axes.

You already know that maths is used in games, for example path-finding algorithms such as the A* algorithm, but it is also used as part of a game's physics. While all objects will have a vector to store their position, some will be marked as physics objects and they will have another vector for velocity, acceleration and mass. These vectors decide the direction in which an object is moving, how quickly it is moving, whether it is speeding up or slowing down and its mass.

Visual styles, graphics processing and editing techniques

The GDD will set out all of a game's design details in terms of how the game should look, and a large part of the design process is dedicated to this.

Worked example: Gravity

Now have a look at how a game would use maths to add gravity to a falling object. Most games will have a game loop function that runs continuously. Every time the game changes frame, the game loop function will run. Some engines call this an 'Update function' or an 'EventTick'.

Now set a numerical value for the gravitational field strength. Both Unreal® and Unity® game engines use a value of about 9.8 m/s² (metres per second per second), so the velocity of the object increases by 9.8 m/s every second in free fall. This is the same as the gravitational field strength on Earth.

Setting the gravitational field strength value to 9.8 and giving an object physics means that in every game loop we would say:

If the object is still falling, keep on increasing the velocity by the gravitational field strength.

Which would mean, while the object has not hit the ground, make it fall faster until it does.

Can you see a fault in this approach so far?

The game loop function loops every time the game changes frame and gravity causes an acceleration of 9.8 m/s each second. This means that the game's gravity is going to be stronger than standard Earth gravity because it is going to increase the falling velocity more often than 9.8 m/s each second. The game's programmers would have to calculate the game's likely frames per second and work out a smaller amount of gravity to add with each game update, or only add 9.8 m/s to the falling velocity after one second has passed.

Visual styles

While there are still a few text-based adventure fans out there, the majority of games are incredibly visual products and their individual art styles and design can often be one of the main selling points for the game. For example Limbo™, by Playdead™, studios is a monochromatic game (black and white) with soft lighting and all the characters are silhouettes (see Figure 8.12). This creates an amazing atmosphere of isolation and tension when playing the game and the art style is so important that every person working on the game had to consider it at all times. This means that the style guide for the game would have been of vital importance, especially to the asset designers.

Figure 8.12: A screenshot from the Playdead game, Limbo™, showing its unusual visual style

Graphics processing

Objects in a game will have a texture or sprite applied to them via a shader. Also known as a material, the shader is a piece of code that defines how the object should be rendered graphically. It will set out properties such as the object's colour, texture, reflections, metallic value or outlines. Shaders are very important for maintaining a consistent graphical style across an entire game. They come into effect during the graphics processing stage of designing a game, before all the objects are rendered into 2D images that are then animated onto the screen, because the shaders decide how the objects should look.

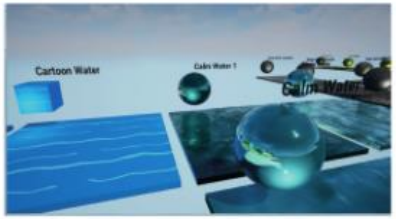

Figure 8.13: Different effects can be created using shaders to allow the game developers to achieve the world designed by the concept artists. Screenshot provided by Ross Everson

Editing techniques

There are many different visual or art styles used in games and this is a major part of what makes games so popular to different audiences. Players may like **cel shaded** games such as Borderlands™, an abstract art style such as The Unfinished Swan™ or a paper craft style such as Tearaway™.

Key term

Cel shaded – is a type of non-photorealistic rendering. It is designed to make 3D graphics appear to be flat. It is often used to mimic the style of comic books.

The art style of the game could be reliant on particular editing techniques. Editing techniques are the tools and features of graphic design software used to manipulate images to get certain effects. This would be the responsibility of texture artists, who may employ a particular technique, such as using a cartoon art style, to create the textures for objects in the game. These textures are created in a professional image-editing software application such as Adobe® Photoshop®.

Platform

Given that platforms vary in terms of capabilities and features, the target platform for a game is a big consideration for the designers. If a game title is intended for one particular platform, then it must be designed to take full advantage of that platform. For example, PC games should take advantage of keyboard and mouse controls as they are the most configurable controllers available and players can decide exactly where they want to map action buttons. Equally, games for the Wii U™ should take advantage of its tablet controller, which has its own touch-screen display that can be used to display extra information and provide additional interaction with the game.

Other titles are designed to be played on multiple platforms. If a game is designed for smartphones, will it work on Apple® iOS, Android™ and Windows® phones? If it does, will it connect to all of a player's game centre social networks and allow players to compare scores or challenge each other? Will it scale up to tablets such as the iPad® or the Nexus® 9? All of these questions have to be answered during the design phase of a game's production. Crowdfunded games may have stretch goals to expand their titles to different platforms after initial design, while games that are funded by mainstream publishers will have set agreements with different platforms before the game is made and they will only expand a game to new platforms if it has done well in sales.

Delivery

Games are designed for either physical or digital delivery, that is, their distribution to customers will either be by means of a physical product (a disc) or digitally (via a download or online streaming). If it is physical delivery, then there will be set deadlines for when the game code needs to be supplied to the factory responsible for printing the discs and game boxes. There will also be an additional amount of time allocated to distributing the game to shops around the world in time for release.

Games that are delivered digitally through platforms such as Steam™ or PlayStation™ Store will be given a deadline for digital distribution but it will not require as many steps to make it ready for the player. The choice of delivery method will often depend on the size of the game and the platform for which it is intended. Mobile games must have digital delivery, as smartphones and tablets do not have disc drives. There is the option for both on PC and consoles. Digital games are usually smaller. However, increasingly gamers like to have the choice of delivery method: a hard copy on disc that they can keep or the convenience of downloading their next game.

Game assets

A game asset is an element of the game that is created outside of the games engine and which has been imported into the engine. The asset may be a visible object that makes up part of the game environment, or it may be a sound effect that plays at certain times during the game. Game assets can be 2D or 3D and they can be graphical, audio or **triggers**. The assets are the building blocks of the game, that is, the game objects.

Key terms

Trigger – invisible collisions in a level that will prompt (trigger) a function or event.

Sprites – images used to represent characters and objects in a 2D game.

Gameplay features

A game's design has to cover all elements of the gameplay features. The GDD should cover all of these elements and ensure that the developers understand exactly how everything should work in a game. The following sections outline the different features of gameplay that are designed for a game.

Interaction model

How is the player interacting with the game? Is it through an avatar, that is, a virtual character who represents the player, or is it through some omnipresence, an invisible hand of god, which controls events in the game and of which the game world's inhabitants have no awareness? The type of interaction sets a variety of design decisions, such as the position and viewpoint of the game's camera, the way information is delivered to the player and how NPCs are interacted with.

Case study

Forest Jump

Consider a 2D platformer game called Forest Jump where a character must jump forever upwards through a forest canopy. The image below shows a scene from the game with labels indicating the types of asset involved.

The character is an asset made up of animated **sprites** that make her appear to jump and run.

The background is a static asset showing the forest background which never changes.

The trees are sprite assets and there are branch assets that have colliders so that the player can land on them.

Each time the player presses the jump button, a springy sounding audio asset is played and when the player reaches the top of the canopy of trees they pass through an invisible trigger asset that tells the game's code that the level has been successfully completed.

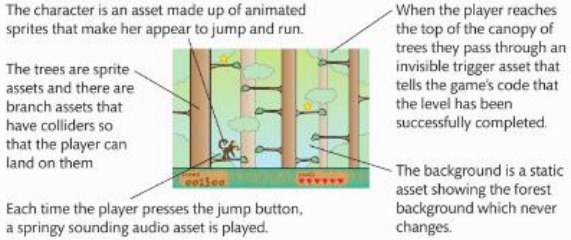

Figure 8.14: A 2D game uses sprites for all its assets

The assets are created using various different tools. 3D assets are created in a 3D modelling software package, such as 3D Studio Max by Autodesk®, and the GDD needs to generate enough concept art for the 3D modellers to be able to create the assets. There is usually a full asset list in the appendix of a GDD, but additional small environmental assets are often added during development as long as they conform to the overall visual style of the game.

Figure 8.15: Examples of 3D assets

Participation

What exactly is the player taking part in? Is it a single player campaign where they follow a story alone? Is it a multiplayer game and, if so, is it a local multiplayer or online multiplayer game? If it is an online multiplayer game, is it a player vs environment (PvE) game where groups of people work together against AI enemies or is it a player vs player (PvP) game where they battle each other in different modes? These decisions will radically change the design of the game. A network programming team is needed for multiplayer games in order to write the code and set up game servers that players will connect to.

Narrative

Does the game have a story? If so, then is it a linear story which will have the same events and outcomes every time, or is it a story with branching narrative where the player's choices affect the outcome of the story? Branching narrative requires multiple scenes and levels to be designed, some of which may never be seen by the player depending on what decisions they make. A story treatment is written as part of the GDD and then, if the game has dialogue, a script will be written for each part of the game. The designers need to decide if the game dialogue is spoken, in which case voice actors must be hired, or onscreen text needs to be produced. While players prefer audio dialogue to having to read text onscreen, it makes it difficult to release the game in multiple countries around the world because, for every different language region that you want to release your game in, all the dialogue has to be translated and re-recorded. This is part of a process called localisation and can be very expensive.

Game setting

Lots of decisions have to be made about the setting of a game and some of these may affect what genre the game is marketed as. The physical setting refers to how the game world is made up. Is it 2D or 3D? How are objects scaled in the world? What are the boundaries and are there invisible walls to stop the player wandering off the path?

Next the temporal setting needs to be decided. When does this game exist in time? Is it in our past, our present or our future? Once that has been established, we need to know how quickly time is going to pass in the game. Is it real time where every game minute is a real-world minute or will time speed up and slow down depending on the events unfolding?

The environmental setting decides where the game is set. Is it in our world or a fantasy world? Is it on a distant planet or inside a microscopic universe? The environmental setting can also dictate what the world is like in theme or culture. It could be a post-apocalyptic world ravaged by zombies or it could be a world under the sea at risk from an environmental threat. Lots of settings in games are repeated from title to title but elements of originality can still be found even in the most clichéd of settings.

We are used to novels and films telling us stories of emotional highs and lows such as love, loss and betrayal. Games are able to take on an even greater emotional journey by letting us make our own decisions about what happens in the story: this is the emotional setting. What can be more heart breaking than watching your character lose the one they love when it was your decision-making that led them to this loss? Games that manage to immerse us in their worlds and combine this with great emotional plots are always well received, and are responsible for moving games forward as an art form rather than just pure entertainment.

Ethical decisions are also a cause of great emotion in games. The designers may choose to give a game an ethical setting by putting the player in a situation where they have to make an ethical choice. For example, it could be between saving one life or a million, or it could be whether or not to perform criminal activities to further a cause but, whatever it is, ethical decision-making can make games incredibly immersive and rewarding. A good example of this is Papers, Please™, a 2D job simulator where you take on the role of an immigration office official checking who is allowed into the country. Once the sob stories start flowing, the bribes are not far behind and the game becomes a series of ethical dilemmas where the player must either risk losing their job for a deserving stranger, or feed their own family.

Goals

The designers decide the main purpose or objective of a game and players will not engage with a game unless it has a point. The designers will decide how the goals are broken down into different levels within the game. Often there is an item that the player has to retrieve or a fellow character who needs rescuing, and this is the goal of the game.

Challenges

The challenges that the player must overcome in order to achieve their goals are the next focus of the designers. What are the actual threats or hazards that the player must face? This stage of the design will see the creation and allocation of NPC enemies to different areas in the game as well as the design of environmental hazards, such as falling floors or jets of fire, which need to be circumnavigated to get to the goal.

Rewards

The designers will come up with a list of potential rewards for the player. One of the most commonly found game rewards are experience points or XP. Players will receive XP for defeating enemies or using a particular skill effectively. Once they have enough XP they may be able to spend it on levelling up or new skills, depending on how the game is designed. Other rewards could be unlockable areas within the game, or, perhaps, additional time or a temporary power-up that can be saved and spent in a future level.

Player actions

What can the player do? Can their character perform basic movements like walk, run and jump? Do they have a more sophisticated set of skills such as magic spells, wielding powerful weapons or teleporting across worlds? All these decisions, while easily made, have far-reaching consequences. For instance, if you give the character the ability to fly, what is to stop them from just flying right to the end of the level over the heads of all the enemies that have been put in their path? Player actions have to be designed to give the player the tools they need to complete the game, but not too easily. Some game designers will not introduce all of the player's abilities at the start of the game, but allow them to be gained, one by one, as the rewards of different levels. There is a style of

game design called Metroidvania where the entirety of the game world is available from the very beginning but some parts are only accessible after certain tools or abilities are won by the player. The name comes from the classic games, Metroid and Castlevania, which had this design.

Rules

Once a character's abilities are decided, the rules of the game must follow. If the player is able to make their character jump, then how high? Can they double jump? Wall jump? The movements of a player significantly affect their progression through the game world, and the movement rules need to be carefully considered. A player's valid moves are the ones that they are allowed to make and the game should ensure that no invalid moves are made accidentally. The rules of a game world do not just apply to the player character, they also apply to enemies, and they decide the physics of the game world as well. If a character has magic, for instance, is it limitless or will they have to top it up with something like mana potions? If that is the case, then how many mana potions should there be per level, and how many can a character hold? All this has to be designed and decided before building the game.

Feedback

The way in which a player is aware of their progress is crucial in a game. The way in which software communicates with humans is a field of study called **human computer interaction (HCI)** and it is never more crucial than in games design. The last thing you want as a player is for your attention to be taken off your object of focus in a game. If you are fighting a boss, you do not want an indicator to start flashing in the middle of the screen and block your view. Games use a head-up display (HUD) to show information in a discrete and unobtrusive way. The more subtly information is fed back to players, the better the feedback, although it does need to be clear.

Key term

Human computer interaction (HCI) – the study of how people interact with machines, and the best possible ways to design interfaces between people and machines.

Difficulty

The difficulty of a game, or degree of challenge, sets out how hard it is going to be for a player to complete the game. Many games use a difficulty curve where the game starts easily and gets progressively more difficult the further through it the player gets. This may be actioned through enemy strength or a number of hazards in a level. Many games allow the player to set the overall level of difficulty at the start of the game depending on their confidence, and some games will release a new difficulty level once the game has been completed on its hardest setting.

Game mechanics

The mechanics of the game include all of the functional game elements that need to be designed and are taught to the player, through either a tutorial level, dialogue or trial and error. Game mechanics may include an inventory system that lets players collect items, a crafting system that combines items to create new ones, or a scoring system whereby different accomplishments generate higher scores and win conditions. A win condition is the rule that sets the circumstances in which the player wins either an individual level or an entire game. In the classic arcade game Donkey Kong™, the win condition is reaching the princess at the top of the scaffold tower. Other titles, such as strategy games, may have more complex win conditions, whereby you must beat the enemy within a certain time frame using only particular resources.

Game structure

The actual structure of the game includes the number of levels, **cut scenes**, enemies and the progression that the player must make to complete the game. This is shown in a number of design documents, such as storyboards which show simple drawings of the sequence of events in a game, flowcharts, diagrams that explain how different rules or algorithms will work, and activity diagrams that show the way in which a player navigates the game from the opening menu to the final credits.

Key term

Cut scene – a cinematic sequence in a game that tells part of the story. Cut scenes can be a separate movie clip or can be shown during a level.

Quality

Games should be designed in such a way as to ensure that they are as high quality as possible, otherwise they will receive poor scores when reviewed and sales will be low. A game's compatibility with its platform needs to be considered during the design phase. For instance, if a game is being made for a touch-screen device, how many different buttons will the player be able to cope with at once? If a game is being developed for multiple platforms, then it must not rely on specific platform features that are not common to all of the platforms it is being designed for.

The performance of a game is also a consideration, as the designers making a game for a device with a lower specification GPU and CPU would not be able to design photorealistic graphics and hours of cut scenes with audio dialogue, because the system would not be able to handle it.

In addition, the designers have to consider the gaming experience at all times during the design stage. Why would people play this game? What is unique about it? How does the level being designed compare to the previous level? Does it have anything new or interesting for the player to engage with? Bad games are games where these kinds of question have not been asked or answered, and where the player is not considered enough.

Design documentation

The game design documentation (GDD) is passed from team to team throughout the games development process. It is of crucial importance and will be referred to constantly by artists, asset creators, programmers and testers.

Audience, purpose and client requirements

The GDD must begin with an overview of the requirements that the client has set out. The client will be the publisher for most games, but for crowdfunded games it will be the backers. Their requirements may include a series of milestones, at certain dates, that the client wants you to meet, or a particular emotion or atmosphere that they want the game to evoke.

The design will then go on to specify the audience requirements, that is, who exactly is the game for and why would they want to play it? Pinning down the target audience is crucial; it is nearly impossible to design a single game that will cater for everyone. There are so many different types of player buying games today and the game designers have to be acutely aware of this. Almost every decision they make when preparing the design document should consider the audience and, because of this factor, it must open with a very clear specification of whom the game is for.

The audience for a game is worked out by looking at various demographics, that is, categories of people based on their age, gender, spending power, education, location and family status. The amount of time that they spend playing games and their gaming experience is also important. You could design a simple platform game in the style of Super Mario Brothers™ for a child based on simple game mechanics, but you could also design a similar game for a 30–40 year old who had enjoyed the original games and wanted to enjoy a nostalgic experience. A game studio might identify its audience based on similar titles that a particular audience has previously enjoyed, sometimes in a bid to steal market share from a competing publisher or platform.

There may also be a specific purpose to a game as well. While most games are designed purely for entertainment purposes, sometimes a developer will create a game that is designed to introduce a new peripheral device, such as a motion controller, educate the player in how to train their brain or learn to play an instrument, or help to advertise a new film or TV series. Creating games for marketing purposes, nearly always creates weak games because their design and development is rushed so that they can be released in time to advertise the new film or TV series that they are tied into. If you wait too long to release a game for marketing purposes, then all interest in that film/TV series diminishes, so games are rushed out to market without being properly designed or tested. It is only when the game is begun at the early stages of a film or TV series' development that it stands a chance of being a quality title.

Case study

A sample client brief

Frodo Games, an international publisher, has asked your development studio to make a game that serves as a prequel to the popular children's TV series *Seb's Teds* which follows the adventures of a young boy and his teddy bears. The TV production company that owns the rights to *Seb's Teds* are letting you use the names and images from the show on the understanding that the game will contain no violent scenes, does not portray Seb or any of the Teds acting aggressively, and maintains the ethics and morals that the show is known for.

- What game mechanics would you consider appropriate for this title?
- Write a description of the audience for this game.

Legal and ethical considerations

There are several laws covering the design and production of computer games and these need to be considered when looking at a game's design.

Copyright

The most important law to consider is the copyright law that protects creative works made by an individual or a company and stops other people from copying them and making money from the copies. The copyright laws cover game content such as characters, places and specific game designs, but the creation of clones in gaming is as old as the games industry itself. Consider the mobile hit Flappy Bird™. As soon as that became popular, it was very quickly followed by Flappy Shark™, Flappy Duck ™ etc, which are clones of the original game with just enough difference that the developers could not be sued for breach of copyright. The trouble is that the law does not protect the look and feel of a game; only the intellectual creation itself is protected so many popular games are cloned and rebranded. However, the developers doing the copying will not have access to the original game code so it will always be an approximation of the original instead of a complete copy. Part of the problem is that it is argued that the functionality of one game is very much like another or similar to hundreds of other titles that have a small element of the game in question. If so many characters run, jump and shoot, would the creator of the first game that contained a character running claim that all other games have stolen their copyright? Games contain lots of elements that fall under copyright protection, for example the images, titles, names and the design of the levels themselves. All this comes under the term intellectual property (IP), which is used to describe the content within a game as well as the game itself. You might hear a publisher bragging about a 'brand new IP' which means a new title that has not had any previous games out and is all-original. A game tied to a film cannot claim to be a new IP, because the film has the IP; neither can a sequel.

Consumer rights

As computer games become a common part of our daily lives, existing laws have been adapted to consider games. One of these laws is the Consumer Rights Act 2015, which gives customers stronger rights as the consumers of (ie the people who buy) video games. UK residents can now claim a refund or repair if the digital content that they have bought, that is the game, is not working or is of unsatisfactory quality. Poor quality is often a consequence of a publisher rushing the development to meet deadlines and not testing the game properly. Interestingly, this does not just cover digital games that need to be paid for, but it also covers disc-based games and games that are free to download. Therefore freemium titles also have to be of a decent quality or the developers are obliged to fix the defective product. This is a good example of the law changing to include new products and patterns in society. Game designers have to stay up to speed with the constant changes in law and the differences between the laws of different countries.

Licence fees

Games designers have to consider other laws when releasing their games and a lot of these are to do with the financial side of games development. If a game is going to be released on a platform such as a console, they will have to pay a licence fee to the console manufacturer to get their game released for that system. Therefore, if a game is made for PlayStation 4™, then the developer would have to pay Sony® a licence fee. This would be paid on the understanding that the game was of sufficient quality to appear on Sony®'s system as the company's reputation would be damaged if inferior products were released. This agreement would be made using a legal contract that sets out the responsibilities and actions of both parties.

Royalties

Another financial consideration is the payment of royalties. If a game is made using a commercial engine such as Unreal® Engine, there may be an understanding that games can be published for free, but, if they earn over a certain amount in profit, then they must start giving a percentage of the game's profits to the commercial engine. Music within a game will also be subject to royalty payments in the same way that it is for a film or TV show. In terms of royalties for the actual developers, they tend to be paid an agreed fee at certain stages of the game's development and, after it has been released, they do not have any claim to the profits. A few games publishers do offer royalties to developers, but it is not commonplace.

Digital rights management

Digital rights management (DRM) is another legal consideration for games developers. This is the area of law that concerns itself with how digital products are used and shared. Once a game has been purchased as a download, is it acceptable to copy this to other devices in the same household or to give copies to friends, or even sell copies? If you bought a DVD of a game you could lend it to friends and later sell it on, but you would only ever have the one copy. Different companies take different approaches to DRM. Apple® will let you set up family networks that share apps, while Google™ lets you log into as many different devices with your Android™ account as you want. Steam™ forces you to authenticate whichever machine you log into with Steam™ and, if another user is on there, you can request access to their game library. Microsoft® originally announced that the Xbox® One™ would have a constant internet connection to check the DRM of any game played. This was introduced so that they could monitor which games people were playing and for how long. They tried to spin it to customers as a way to get constant updates, but this received such a backlash from players that they quickly changed their mind. Their competitor Sony® introduced a 'virtual couch' that lets friends play each other's games whenever they are both online.

Ethical considerations

No specific law covers the ethical considerations of games design, but it is something that games designers have to think long and hard about. If they are representing a particular country or culture, are they doing it in a way that is ethically acceptable? (For example, they should not suggest that all the people from a particular religion are violent. They cannot discriminate or portray stereotypes in games.) If they are putting the player in a questionable situation, are they clear that they appreciate that this is an ethical question and not something that the player should just accept? Questions about violence and sex in video games are always present in the media and many games have been accused of provoking all sorts of real-world crimes. Therefore games designers have to think very carefully about what they are depicting in their games and who could be influenced by them.

PAUSE POINT

Have a look at the blog www.gamerlaw.co.uk

Now imagine that you have created an amazing new game that is unique in its story, features and visual style.

Find out how you would be protected by copyright law.

Hint Copyright law protects 'the expression of an idea' but not 'the idea itself'. How will this protect you?

Extend What will happen if you release your game in different countries?

Game design

The following features should be included in the design documentation for a game.

- Type of gameplay – a definition of what the game involves, eg FPS, MMO, RPG.
- Data dictionary – a table showing all the data used in the game, its type (eg number or text) and its purpose.
- Algorithm design – designs for the game's functions and features using **pseudocode**.

Key term

Pseudocode – a way of writing code without it being in a specific programming language.

- Storyboards, flowcharts and an activity diagram – all show the structure of a game, how the game should look and flow.
- Visual style:
  - world – the type of terrain, style of architecture and style of objects
  - characters – player, enemies and other NPCs
  - feedback interface – how information is communicated to the player
  - perspective – 2D, 3D, first-person, third-person, scrolling, aerial, context-sensitive.
- Full motion video – used for cut scenes and menus.
- Asset lists – graphical, audio, video.
- Gameplay features – what features make this game original and unique.

Link

For more on gameplay features, see the section on Gameplay features.

Worked example: Hazman

Here is a sample of the game design for a strategy game called Hazman. (The full design documentation would be a lot longer and more detailed.)

Type of gameplay – a point and click strategy game for PC that sees a young hero clearing chemical spills in industrial areas around the world. As the game goes on, Hazman starts to suspect that these incidents are not accidents after all and a conspiracy theory soon leads to a dramatic rescue.

The player will quickly assess an area and race against the clock to find the right solutions to clearing up the different spillages. The gameplay is a combination of solving coded puzzles by mixing the right chemicals together and 'escape-the-room' style puzzles to get the hero and any good NPCs to safety.

Data dictionary – each level will have the following data requirements.

Name	Data type	Data length	Scope	Purpose
LevelNumber	Integer	1–99	Public	Stores the current level number that the player has reached.
MovementSpeedMax	Float	0–25	Public	Stores the current movement speed based on input from player.
PlayerScore	Integer	0–4	Public	Stores the current score of the player.
ChemicalTypes	Integer	0–9	Public	Stores the amount of different chemicals that can be spilled in current level.
ExitPosition	Vector3	(0–999,0–999,0–50)	Public	Stores the position of the level exit.

Algorithm design – the game level will be split into tiles, with 4 rows by 4 columns. Each spilled chemical will start on one tile and, as the game progresses, will spread onto more tiles until the player becomes stranded.

Pseudocode:

```
LevelGridArray{0000,0000,0000,0000};
On Game Start
  While ChemicalTypes>0 - Set random grid element to 1, ChemicalTypes--
On Game Loop
  For Each Chemical - Spread to next available empty grid space till array full
```

Storyboards – the opening cut scene for the game will be a series of still cartoon panes that tell the story of the first chemical spills. Additional cut scenes will be made after release, available as downloadable content.

Figure 8.16: A storyboard is a visual plan of how aspects of the game should look, usually showing cut scenes

Visual style – the game's style will be highly stylised cartoon visuals with thick black outlines around all assets. All floor tiles will have simplified textures and the chemicals will have a unique colour that is bright and has a shiny surface. The characters will only be seen from above and they will have exaggerated proportions (large heads and eyes, a small torso, short limbs and large feet). The HUD will display the time prominently on the top right and messages will be presented to the player in the centre of the screen (gameplay will pause until the message is discarded by the player).

The game will have a top down perspective throughout gameplay where the player clicks on different tiles to activate the puzzles and solutions. Where a solution is a success, a **context-sensitive perspective** will zoom in on the player's character whose animation will change to celebration mode.

Full motion video – there will be a cut scene at the start and end of the game. This will be an MP4 containing static images and the game's theme tune.

Asset Lists

Graphical	Audio	Video
Player sprites	Level specific theme tune	Opening cut scene
Floor tile sprites	Spillage Sound Effect (SFX)	Closing cut scene
Wall sprites	Creeping chemical SFX	
Hazard sprites	Death SFX	
Crates sprite	Win SFX	
NPC sprites	Time running low SFX	
Chemical sprites	Countdown SFX	
Menu textures		
HUD textures		

Gameplay features – the game is puzzle based. The first four levels will contain chemical puzzles where each chemical has an antidote, made up of ingredients that Hazman must collect, which will be spread around the level. Once these ingredients are mixed together, the level will be complete. The next four levels will involve escape-the-room puzzles where there are no antidote ingredients, but Hazman must assemble items to escape from the room. For instance, he will need to get a key from a monkey in a cage, and, once he opens the cage with a crowbar, the monkey will jump to a higher shelf out of reach, Hazman must then find a banana to tempt the monkey down and retrieve the key.

Key term

Context-sensitive perspective – this is when the in-game camera changes to focus on something that the player has selected or a change that they have caused.

Development choices

There are many different decisions that the development team need to make as they embark upon the creation of a computer game.

Platforms and programming languages

A number of different factors decide how a game is made and one of the most crucial of these is which platform it is being developed for. If the developer knows in advance that the game is only going to be released on one specific platform, then they will be able to find out the language that the platform is designed to be used with along with all the different games engines that can be used to create games for that platform. If they know that they are going to release the game onto multiple platforms, then they must use a games engine that supports all of these different platforms, or risk increasing the development time, because they would have to start from the beginning for each intended platform.

If you are writing games purely in code, then an integrated development environment (IDE) is used. This is a system, such as Visual Studio®, which supports one or more programming languages and is able to find errors in the program and, once finished, package the code into an

executable application. A games engine will use an IDE to write the code portions of the game or it will have the IDE built in. For example, a console game that is going to be exclusively for the PlayStation 4™ could be written in C++ and all of the input and output will be designed specifically for the PS4™. It could be written in the Visual Studio® IDE. It is not possible to build levels and arrange assets visually in an IDE alone, which is why games engines are used.

If a game were going to be released on PC, Mac® and Linux®, then it would be best to create it on a games engine that supported all three platforms as it would be difficult to write the game code to work on three different operating systems directly in code.

Some programmers are able to use a simple text editor to write code and make games. In the 1980s, it was possible to buy magazines which contained code samples for players to copy to create their own games on systems such as the ZX Spectrum or Commodore 64.

C++ is the most widely used programming language for games. C# is a derivative of C++ but it is missing some features and does not run as fast. However, it is very useful for writing game mechanics for engines such as Unity® because it is easier to understand, but still powerful. Java is another object-oriented language, which can run slowly due to the way it is translated for any platform. It has not been embraced by games console developers for this reason, but it is the best way to write programs for Android™ without using a games engine, because the Android™ system is based on Java. Scripting languages like JavaScript and HTML5 are widely used for making web games because they are quick to run and small in file size. Older languages such as Python™ or Delphi® are not really used in professional games development but are very useful for learning how to program.

Application programming interfaces

The design of the game should also include reference to any application programming interfaces (APIs) that are going to be used. An API is a code library that has been written for use in other programs in order to quickly add extra functionality. It may be that the game needs to communicate with an external service, such as Apple®'s iOS Game Center for Developers to take advantage of online leaderboards, or it could be that the developers want to add a virtual reality option so the Oculus Rift™ API might be used. When a programmer wants to use an API, they reference it at the top of the code where it is needed in an 'Include or Using statement'. After the API has been included, the programmer can reference it whenever they want during development.

Computer game development kits

If a game is being designed for PC or web, then it is quite straightforward to test the code during development and the designers will ensure that the correct hardware is present before the build begins. If the game is being built for a console however, the development team will need to buy a console development kit for whichever platform they are designing it for. A console development kit is a version of the console that can connect to a PC and have test builds of the game deployed directly onto the console, to test how it works on the system for which it is being designed. This is the best way to test a game for a console and it ensures that all of the controls work, that the game loads and saves correctly and that any communication with the console's online capabilities work correctly. Some console manufacturers sell development kits while others are willing to lend them out to developers, especially ones they have worked with before.

Intended platform/media for delivery

An important question for developers to ask themselves considers which features of the intended platform the game will take advantage of. You may want to keep all the game's features common to all platforms or have specific functionality designed for individual platforms. These features should be included throughout the game, and should also be promoted in the marketing for the game as unique selling points.

In addition, the intended rating of the game needs to be considered during the design process (see Table 8.3). If a game is going to have a PEGI rating of 7 then it may have violence but the violence must be unrealistic (ie something that could not happen in real life, like a magic spell or a piano falling on someone and them recovering instantly). The rating makes a huge difference to a game's design and everyone involved in the design needs to know what the intended rating is so that they can make their designs age appropriate.

Another consideration is whether a game will be delivered as a digital download or a boxed game. If it will be released as a digital download, then there needs to be a series of graphical assets created for the game's entry on the download page. Digital platforms will also ask for a number of differently sized icons: these are graphics that can be resized, and graphics that can be used as part of a marketing promotion or sale (boxed games also need graphics for online advertising). All of these images have to be provided by the game's development team. If it is going to be a boxed game, then it also needs to have box art drawn up, an image to go on the printed disk and any instructions or promotional codes that are going to be included inside the game box.

Table 8.3: PEGI games ratings, for more information visit www.pegi.info

3	Suitable for ages 3 and older. May contain very mild violence in an appropriate context for younger children, but neither bad language nor frightening content is allowed.
7	Suitable for ages 7 and older. May contain mild or unrealistic violence (eg violence in a cartoon context), or elements that can be frightening to younger children.
12	Suitable for ages 12 and older. May contain violence in either a fantasy context or a sporting action, profanity, mild sexual references or innuendo, or gambling.
16	Suitable for ages 16 and older. May contain explicit or realistic-looking violence, strong language, sexual references or content, gambling, or encouragement of drug use.
18	Unsuitable for persons under 18. May contain extreme or graphic violence, including 'violence towards defenceless people' and 'multiple, motiveless killing', strong language, strong sexual content, gambling, drug glamorisation, or discrimination.

Animation timeline

Different parts of a game will involve various different animated sequences. Animated characters will have a series of animation cycles that they will play on repeat when the character performs different actions. For example, walk cycles, run cycles, idle cycles and so on. The artists will draw single frame images of each position needed in the animation.

Cut scenes within a game involve assets used within the game, so the artists will create storyboards that follow the cut scene and may create a timeline showing the order in which assets are needed for that scene.

Production schedule

The schedule for a game is recorded in a number of different ways. The project milestones are set up first. The milestones are a list of stages that the project will go through, and the dates by which they need to happen. The art stage is followed by asset creation, mechanics implementation, level building and testing. Completion of these milestones may trigger the publisher to pay the development studio a portion of their fee, while any delays could result in a reduction in payment.

The GDD will contain a more detailed breakdown of the schedule. This may be a Gantt chart, which is a kind of bar chart that displays all the different tasks that need to be carried out to complete the project, who is doing them and when they need to be completed. The team may use project management software to create and share the schedule. It is usually a live document that is regularly updated and adapted based on changes during development.

Resources

Games cannot be designed without various pieces of hardware and software and this can be one of the biggest expenses for a games studio, especially if they are making a game for the first time. Every developer in the team will need access to a PC and the appropriate software. 3D modelling software is very expensive, as is graphic design software, but they can be purchased on a monthly basis which means that the studio only has to pay for what they need. However, this is not a good long-term approach. The PCs required for games development need to have a good enough hardware specification to be able to model and animate in 3D.

Graphic designers and concept artists will need additional resources such as graphics tablets in order to complete their tasks (see Figure 8.17). They will also require traditional drawing equipment (pencils, pens and paper) and scanners to make these images digital so that they can be shared.

Figure 8.17: A graphics tablet is an invaluable tool which provides a lot more accuracy than a mouse

In addition to these resources, a games studio will need a building to work in, and they will need to pay for utilities such as electricity, gas and broadband internet to function as a business.

Test plans

After a game has been created, it must be tested thoroughly. Players will not be happy if they have paid for something that does not work properly. Testing is taken very seriously and, while a lot of people imagine it to be tremendous fun to play games for a living, professional games testers need to be able to understand both the design documentation and the programming code because this is what they will use as a reference when

they are creating and executing their tests. The test plans they write need to check the playability, performance and quality characteristics of the game.

There are a number of different strategies that can be used when planning the testing of a game, and the producer or project manager will be responsible for employing a test manager who will then choose the most appropriate tests. There are various types of testing documentation that can be used to plan the tests and then record the results.

The test plan covers the development of the entire game and makes sure that there are no flaws in the product that is eventually released. This is a very large-scale task and if any problems are found during testing the test plan will ensure that they are recorded and fixed. Games that are released with flaws or errors in them do not sell very well and receive poor reviews from games journalists.

Each test will have a test case, which is a particular scenario that the tester will undertake to try out a feature of the game, or a command, under certain conditions. These conditions might be the direction of movement or the number of items in an inventory. It is important that the test cases cover as many different scenarios as possible so that there is nothing that the player could do when the game is sold that has not already been tried during a test.

The test plan is used as a test log, where all of the outcomes are recorded and an indication of what needs to be done to correct any errors is added. If a problem is found, the tester may take a screenshot to use as test evidence. This screenshot is then passed on to the developer who will use it to get a clear idea of the problem and then fix it. Once all of the testing is complete, a test report is written to show where problems were found, what has been fixed, and to summarise any vulnerabilities in the software or issues that might resurface. Once a game has been tested to a satisfactory level, it will then be released for sale.

Case study

Working out mesh constraints

The official documentation for Unreal® Engine 4 states that the maximum vertex count for a level in a mobile game is 65,000 vertices per mesh. So what does this mean? A 3D object in a game is created in software such as Autodesk® 3D Studio Max and is referred to as a mesh. Meshes are made up of single points called vertices and every vertex in a virtual world has to be calculated and positioned. If a simple cube mesh has 8 corners, then it has 8 vertices. If you divide the maximum possible amount of vertices, 65,000, by 8, you get 8125. This means that you could only have 8125 cubes in a mobile game.

It does not seem like that big a deal until you consider that a typical console character can be over 80,000 vertices: this is too many for a mobile game. Any character in a game will instantly increase the vertex count, especially if they are animated, as this requires more vertices in order to stop the mesh looking strange when it moves.

The next consideration is the number of bones used in animated characters. A standard human skeleton rig in an animated 3D character has 60 bones. The upper limit for bones in a mobile game according to the Unreal® Engine 4 documentation is 75 per mesh. This means that if you use the typical skeleton rig for animated characters, then you may not have enough bones left to have any other animated characters in the game.

Therefore designers working on mobile games will do a number of things to make sure that they can have multiple animated characters in their games. First, they will design the game with a very basic art style so that the meshes use as few vertices as possible. The more curves an object has, the more vertices it will require. So designers will create objects and characters with as many straight edges as possible. If you look at 3D games on old consoles, you will notice how the designers have avoided creating curves because they were limited in the number of vertices they could have.

Developers will also use level of detail (LOD) groups. This is a process where a simpler version of a 3D model is put into a game level when it is far away from the player, but, when you get closer to it, a more detailed version of the mesh is swapped in. For example, the less detailed version of a house may be a flat cube with images of windows and doors, but, when the more detailed version is swapped in, the house has doorframes, handles, window ledges and window frames.

Finally, developers can design their characters so that they are either not animated at all (they could still rotate or move), or they have simple skeletal rigs (no finger bones or feet bones, for example). Characters in mobile games tend to have cube-shaped feet and hands, which developers tend to get away with because these characters are viewed on small mobile device screens.

Constraints

The constraints of a platform are the limitations that designers have to consider when producing the design documentation. These limitations can have far-reaching effects on how the game is going to play.

Platform limitations – hardware

Perhaps the biggest concern for designers is what the hardware can cope with. A high specification gaming PC with an Intel® i7 processor, 16GB of RAM and a powerful graphics card will not have any difficulty running any game that is thrown at it but if a game is being designed for a less powerful platform then the abilities of the hardware will directly affect the game design.

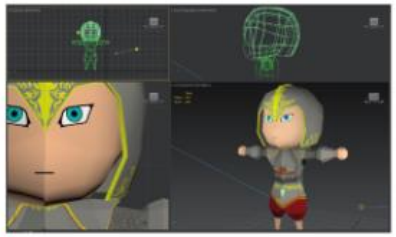

Figure 8.18: A 3D Character designed for animation. Screenshot provided by Dan Bennett

Platform limitations – software

The software on a system does not produce as many constraints to games designing as the hardware does, but the version of an operating system that is being used may cause some problems. Mobile operating systems are updated all the time and if a game is not compatible with a few older versions then the designers will lose a portion of their target audience. Apple® fans often complain that new system updates make their hardware redundant and force them to upgrade, and that this also reduces the number of games available to them because a newer game, marked on the app store for a higher iOS version, will not even appear to them.

Reviewing and refining designs

Producing the design documentation for a game is not something that you do once per game and then make no changes. A game's design should be reviewed as often as possible, and the more time set aside in the schedule for looking at the designs and for seeing if they can be improved, the better.

Primary feedback

After the initial design documentation is completed, a game's designs will be shown to clients and other interested parties, potential players or any licence partners, to see what they think of them. They will be asked about the quality, effectiveness and appropriateness of the designs for the intended audience. Sometimes, a number of **prototypes** are made as part of the design process and these will be demonstrated to potential users via the internet or at gaming conventions. Designers have to be careful about showing too much too soon because a lot can change during the design stage. However, much can be gained from knowing what your potential audience thinks about your ideas. It provides you with feedback with which to review and refine your designs.

Key term

Prototypes – small test game levels used to make sure that the key features of the game are working and to illustrate to clients and potential customers what a game will be like.

Client communication

Prompt and professional communication with the client is crucial for the design team to show that they are on top of the project and that they are not hiding any slipped deadlines or major problems. Whether by email for brief exchanges, or face to face with the client for important discussions, the design team has to be in regular communication with their client. Many games studios subcontract some areas of the design and development to different studios and this can cause a breakdown in communication which may result in a weak final product. In 2013, a highly anticipated game received very low review scores and disappointed the gaming public because it did not contain scenes that had been in the promotional material. A group of gamers took the developers and publishers to court. The publisher blamed the developer but the developer claimed that it was the responsibility of the publisher. It is possible that these problems could have been avoided if there had been better communication between the developers and the publisher.

Meetings and timescales

Whenever different teams within a developer are working together, and when they are working with a publisher or subcontracting elements of the development, it is important to have regular meetings and to keep records of these meetings in the form of minutes, decision logs and action lists. In the games industry, one form of **project management** used is called Agile Scrum. This approach

requires the teams working together on a creative project to meet every day for 'daily scrums' where they discuss the current progress and identify any changes that need making.

Key term

Project management – different methods and procedures used to keep a project on track and under budget by minimising and mitigating risks and issues.

When changes to the designs are made, either due to feedback or because of review work within the design team, the timescales for the overall development must be updated to include how long the new changes will take to implement.

Updating the design documentation based on feedback and reviews means that a new copy must be distributed to the entire development team as soon as the changes are approved. Alternatively, a live document can be edited which everyone has access to.

Refining ideas and solutions

When the design for a game is started everything is possible, but then, necessarily, it is slowly refined down into a more specific and sophisticated design that players will want to play. Through making prototypes, it is possible to see which ideas work or are popular, and then to refine them down to the best possible version. The game's designers will choose their refinements by selecting the most successful (popular or effective) features, and by removing elements that proved too difficult to get working properly in the prototypes. The game's publishers and potential customers will be shown the prototypes and asked for feedback, which will then inform their ideas. Problems in the game's design that are revealed by prototypes require solutions which, once found, are then included in the design refinements. It is hard to find out which features were cut out of popular games as the games studios tend to keep these details to themselves, but the internet is full of rumours of features that might have been.

Theory into practice

Media and communication skills

The games industry is at the forefront of creative media and technology. Given this, its working practices tend to be driven forward by dynamic individuals and cutting edge software. You have to be able to use excellent communication skills to ensure that your role in the game's design and development is as efficient and effective as possible. Below are lists of the written and verbal skills required to work successfully in the games industry.

Written skills:

- be able to use email to share or request information
- create design documents that colleagues can easily interpret
- write reports detailing a project's progress for clients
- create presentations with visual aids to assist understanding.

Verbal skills:

- be able to communicate effectively one to one with colleagues and subordinates
- be able to communicate and work effectively as part of a team
- be able to communicate effectively in both informal and formal situations.

How effective your verbal skills are will depend on how well you use tone and body language, as well as what you actually say, to remain professional and convey information. This is especially important when you are giving presentations or talking to clients. You have to use positive language and be able to reassure your audience that you are in control and you understand the scale and scope of your presentation topic. You must use the appropriate technical language when talking to people, especially when presenting information across different teams. If you use too much technical jargon with non-technical colleagues, the discussion may be lost on them.

1. You must always consider how you are responding to people. Are you being supportive? Are you making sure that everyone is getting the chance to talk, not just the loudest person at the table? If you are good at resolving arguments or conflict in teams, or managing the expectations of clients, then a leadership role in games development could be in your future. Make a list of ways in which you can develop your written communication skills.
2. Write a list of tips for yourself on how to communicate effectively in the following situations:
   - when discussing the design of a game with teammates
   - when delegating design tasks to subordinates (people that you manage)
   - when presenting the design of a game to clients.

Write down ideas for how you would be supportive to teammates and subordinates in a design team and how you would make sure that everyone's opinions and ideas were heard.

C Develop a computer game to meet client requirements

Programming a computer game can be very different to creating an office software application or a web application but there are still similarities.

Just as a software application uses code to create menu structures and events, so does a game: it can dictate the flow of the game from the loading screen, and it accesses files in similar ways for game saves. When games programmers are deciding how to proceed from the design documents that they have been given, they usually have the first decision made for them: which programming language they should use. This decision tends to be taken out of their hands because the choice of platform will decide how the game is going to be built. There are still some choices available to the programmer but they are narrowed down a lot. Some computer games development companies will have a particular house language that they insist that everyone uses. The implementation of the programming will depend on the language that is being used but also on the complexity of the game.

Principles of computer games development

Before development begins, there is a basic understanding shared amongst the development team about the principles of games development. When working on a game all team members, regardless of their individual role, have to understand every step of the process in order to ensure that they are working at the optimum level.

The design documentation should have been distributed to the entire team before the development phase. Certain technical diagrams, also known as **schematics**, are delivered to the people who are responsible for the production of those elements. If a 3D game has a third-person character, then they will require a 3D mesh that has been rigged, a series of animations, a state machine to control the transitions between animations, space and a player control program that reads input from the player and changes the animations, movement and rotation of the player character accordingly.

Computational processes

Various computational processes are used as part of games development. These processes happen discretely in the background of the development process and during runtime when the game is playing. One example of this is the **rendering engine**, a system that is responsible for changing the massive data set that represents the objects, textures and players in a game and converts them into a series of 2D images that are rapidly changed. We believe we are viewing a world in 2D or 3D and seeing it move but, as with TV, film and traditional animations, we are watching a series of flat images being quickly animated before our eyes. That is why, when a system slows down or **lags**, the images can freeze (we are stuck on one image waiting for the next one to render).

The screenshot below (Figure 8.19) shows how a rendering engine can also add effects on top of the world that we are

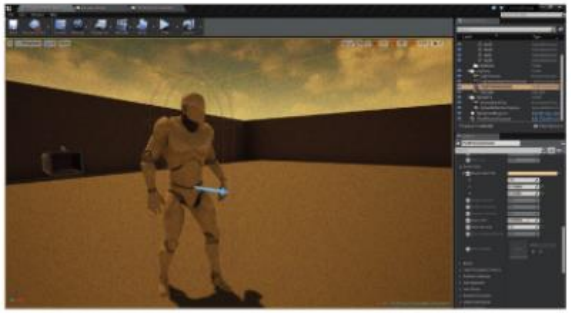

Figure 8.19: Camera effects can be added to a game

Step by step: Production of a third-person 3D game using a games engine

4 Steps

1 The **3D modeller** is given concept art and detailed designs for the player character. As the character seen most in a third-person perspective game, it will have lots of vertices allocated to it. The modeller will understand, based on the animation schematics, how the character will need to move so they will create the mesh with movement in mind. This means that when the animator takes over, they will receive a model that has the rigged bones ready for animation and is modelled in such a way that allows for all the required animation. Imagine that a human character had been modelled and nobody had told the modeller that one of the animations sees the character reveal a tail that they had been hiding. If the tail was not modelled and rigged, it would not be able to feature in the game.

2 The **animator** will use the animation schematics to create all of the different **animation cycles** that have been requested. In **photorealistic games** with big budgets, the animators will use **motion capture** studios, where actors perform the animation movements while wearing motion capture suits. This speeds up development but costs a lot of money.

3 The animator will then hand the animated rigs over to a **technical artist** who will put the animated character into the games engine and create a state machine, which is a system that decides when one animation should change into another, for example when a character should go from standing to walking and from walking to running. The technical artist does not need the animator to explain all of the animations created to them because they can see them in the design documentation and perform their part of the job easily.

4 Once the animations have been completed by the technical artist, they are handed over to the programmer who has been writing code to make the animations change on the push of a button or pressure on an analogue stick. This example is for the production of a third-person 3D game using a games engine. The process would differ depending on the type of game and size of team.

Key terms

Schematics – a technical diagram showing the content and function of game elements.
Rendering engine – the software in a games engine that converts virtual worlds into 2D images which are then animated.
Lag – a delay or reduction in the game's frame rate.
3D Modeller – a job role that involves the creation and texturing of 3D objects.
Animator – a job role that involves the creation of movement in game objects.
Technical artist – someone who works between the technical and design teams and understands both.
Animation cycles – different motions that characters will use on repeat such as running and jumping.
Photorealistic games – games that try to look as lifelike as possible.
Motion capture – a process where real-life motion is converted into data that can allow game characters to move in the same way.

being shown. It might be a temporary effect tied into the game's story or an artistic decision that is made later, but post-processing effects can be added to change the overall look and feel of a game. The example in the screenshot shows a film-grain effect being added to make a game seem more 'old world'.

Physics engines are an important computational process that can affect the speed at which a game runs. If a game uses too many physics objects, then the game can slow quite significantly.

Lighting is another crucial computational process and the real-time creation of shadows is a process that can slow a game down significantly or require that it is only played on a very powerful machine. Some PC games allow you to change the quality settings in order to run a game on a slower specification. One of the first things to get turned off will be real-time shadows. As with game physics, something that happens easily in the natural world takes a massive amount of calculations and processes to simulate in a game world. Shadow quality can, usually, be changed. A hard-edged shadow looks unrealistic in most situations, but more realistic than an object that does not cast any shadows. Soft-edged shadows that fade into the light are a lot more realistic, but require the shadow to be rendered a number of times and therefore take a lot more computational power.

Research

The next time you play a game, pay attention to the shadows and physics in the game world and see if you can notice where developers have taken short cuts. For example, see if the main character has a shadow. Can you see any shadows that are not affected by the main source of light in a game?

Applying mathematics

When a game is being developed, all the objects which require physics will have it applied. This is usually done in a games engine but, when the game is written just using code and assets, the physics will be an API which is then applied to objects that require physics in the code.

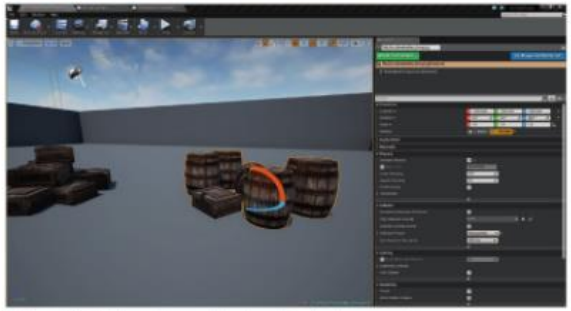

Figure 8.20: The barrels have got physics activated in the game engine

The screenshot above (Figure 8.20) shows how a games engine applies physics, with a simple tick box and then the application of various different settings, such as how much mass the object has. The image also shows the three different vector values that are applied to the position, rotation and scale of the object. These vectors all contain three values which correspond to the *x*, *y* and *z* coordinates of 3D space. The object is selected using the rotate tool and each of the three axes are presented as a direction in which it can be rotated.

The application of maths while the game is running is dependent on the features of the game. One common maths function that is applied to game objects is called lerp, which stands for linear interpolation. This is movement from one position to another and is used on moving platforms, simple enemies and health bars, for example.

Prototypes and engines

If the development is taking place in a games engine, the correct choice of engine must be made. There are many to choose from and, as most of them include a purchase price or payment of royalties, it is important to get the choice correct from the start. Two of the most popular engines, Unity® and Unreal® Engine, have achieved

Step by step: Production of a third-person 3D game using a games engine

4 Steps

1 The **3D modeller** is given concept art and detailed designs for the player character. As the character seen most in a third-person perspective game, it will have lots of vertices allocated to it. The modeller will understand, based on the animation schematics, how the character will need to move so they will create the mesh with movement in mind. This means that when the animator takes over, they will receive a model that has the rigged bones ready for animation and is modelled in such a way that allows for all the required animation. Imagine that a human character had been modelled and nobody had told the modeller that one of the animations sees the character reveal a tail that they had been hiding. If the tail was not modelled and rigged, it would not be able to feature in the game.

2 The **animator** will use the animation schematics to create all of the different **animation cycles** that have been requested. In **photorealistic games** with big budgets, the animators will use **motion capture** studios, where actors perform the animation movements while wearing motion capture suits. This speeds up development but costs a lot of money.

3 The animator will then hand the animated rigs over to a **technical artist** who will put the animated character into the games engine and create a state machine, which is a system that decides when one animation should change into another, for example when a character should go from standing to walking and from walking to running. The technical artist does not need the animator to explain all of the animations created to them because they can see them in the design documentation and perform their part of the job easily.

4 Once the animations have been completed by the technical artist, they are handed over to the programmer who has been writing code to make the animations change on the push of a button or pressure on an analogue stick. This example is for the production of a third-person 3D game using a games engine. The process would differ depending on the type of game and size of team.

Key terms

Schematics – a technical diagram showing the content and function of game elements.
Rendering engine – the software in a games engine that converts virtual worlds into 2D images which are then animated.
Lag – a delay or reduction in the game's frame rate.
3D Modeller – a job role that involves the creation and texturing of 3D objects.
Animator – a job role that involves the creation of movement in game objects.
Technical artist – someone who works between the technical and design teams and understands both.
Animation cycles – different motions that characters will use on repeat such as running and jumping.
Photorealistic games – games that try to look as lifelike as possible.
Motion capture – a process where real-life motion is converted into data that can allow game characters to move in the same way.

being shown. It might be a temporary effect tied into the game's story or an artistic decision that is made later, but post-processing effects can be added to change the overall look and feel of a game. The example in the screenshot shows a film-grain effect being added to make a game seem more 'old world'.

Physics engines are an important computational process that can affect the speed at which a game runs. If a game uses too many physics objects, then the game can slow quite significantly.

Lighting is another crucial computational process and the real-time creation of shadows is a process that can slow a game down significantly or require that it is only played on a very powerful machine. Some PC games allow you to change the quality settings in order to run a game on a slower specification. One of the first things to get turned off will be real-time shadows. As with game physics, something that happens easily in the natural world takes a massive amount of calculations and processes to simulate in a game world. Shadow quality can, usually, be changed. A hard-edged shadow looks unrealistic in most situations, but more realistic than an object that does not cast any shadows. Soft-edged shadows that fade into the light are a lot more realistic, but require the shadow to be rendered a number of times and therefore take a lot more computational power.

Research

The next time you play a game, pay attention to the shadows and physics in the game world and see if you can notice where developers have taken short cuts. For example, see if the main character has a shadow. Can you see any shadows that are not affected by the main source of light in a game?

Applying mathematics

When a game is being developed, all the objects which require physics will have it applied. This is usually done in a games engine but, when the game is written just using code and assets, the physics will be an API which is then applied to objects that require physics in the code.

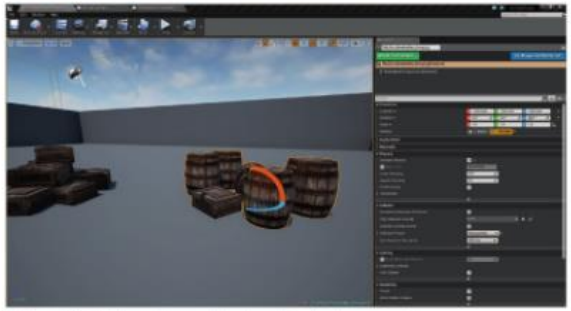

Figure 8.20: The barrels have got physics activated in the game engine

The screenshot above (Figure 8.20) shows how a games engine applies physics, with a simple tick box and then the application of various different settings, such as how much mass the object has. The image also shows the three different vector values that are applied to the position, rotation and scale of the object. These vectors all contain three values which correspond to the *x*, *y* and *z* coordinates of 3D space. The object is selected using the rotate tool and each of the three axes are presented as a direction in which it can be rotated.

The application of maths while the game is running is dependent on the features of the game. One common maths function that is applied to game objects is called lerp, which stands for linear interpolation. This is movement from one position to another and is used on moving platforms, simple enemies and health bars, for example.

Prototypes and engines

If the development is taking place in a games engine, the correct choice of engine must be made. There are many to choose from and, as most of them include a purchase price or payment of royalties, it is important to get the choice correct from the start. Two of the most popular engines, Unity® and Unreal® Engine, have achieved

their success by being able to port games onto multiple platforms, whereas other engines such as Cocos2D-x™ are aimed at one particular platform.

Figure 8.21: Game mechanics should be tested in small prototype levels

Once the games engine has been decided upon, a number of rapid prototypes will be created to test and show off some of the game's unique features. These prototypes will be used to check the feasibility of a feature, explore the timescales needed for development or to solve problems and to identify issues early on. The prototypes will sometimes be used to create trailers or promotional materials for the game, but developers need to be careful because players have long memories and if the final product seems too different from early footage then complaints will be likely.

Tools and techniques for development

There are many different tools and techniques used in games development, and the more mature the industry becomes the more creative developers are getting.

The image below (Figure 8.22) shows landscape sculpting, which is a technique available in many games engines whereby a large flat mesh is created and the developers are given various sculpting tools in order to create a realistic landscape.

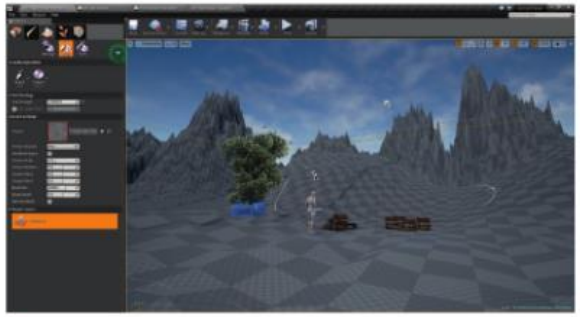

Figure 8.22: Landscape sculpting can save huge amounts of development time

A multi-layered landscape material will then be applied which allows the environment designers to colour in the landscape with different textures that will represent, for example, rock, sand and grass (see Figure 8.22).

Another technique is high-to-low poly modelling whereby a 3D modeller will create a highly detailed version of a mesh and then create a texture from the model that is then applied to a low-poly mesh. Doing this creates the impression of detail without having to render complex shapes.

Choosing the right tools and techniques depends on the type of game in development but also the experience of the development team.

Quality assurance

The quality assurance (QA) process begins during the game design with the designing of the test plans, and as development begins the testing team will start to grow. The testing process is not just about finding problems – it is also about suggesting potential fixes and refinements. Testers are more commonly known as QA technicians and it is a very responsible role that requires people with meticulous attention to detail. Testers should not just be people who enjoy playing games, but people who are able to think of all of the different ways that a game can be broken, and are able to follow formal processes to document every problem that they find in a way that can be understood by the programmers.

The testing is split into different phases and begins with unit testing all of the individual technical elements of the game. This is followed by alpha testing the first complete build, beta testing the next release after bugs have been fixed and then a final quality test to ensure that everything is working. Sometimes the developers will offer a beta build to the public for testing, as this gives them the chance to get much more feedback.

Technical constraints

While we have already looked at the design constraints that can affect how the game will look and feel, a number of technical constraints may affect the development stage. The game will have a set budget and this will dictate how much time can be spent on development, as most of the team will be on fixed-term, temporary contracts. Players often ask why games do not have all the features it is possible to have and budget constraints are one of the biggest reasons for features being cut back.

The size of the servers will affect the development of an online multiplayer game. The number of players who can access the game online at one time is known as the maximum concurrent players. If the server is unable to cope with all of the potential players across the world, then the workload may be split between multiple servers for set areas such as, for example, Europe and North America.

Developing computer games

In this section, we will follow the step-by-step development of a third-person game called Jump Chase where the player has to race through different levels to get to the end goal within a set time limit. Known as a platformer or platform game, this style of game has always been a popular design.

Worked example: Creating Jump Chase

Visual style

As a platform game relies on precise jumps and timing, the perspective works best with a third-person avatar so that the player can judge all their jumps carefully. The game is being made in Unreal® Engine 4 and, as the screenshot below shows, the sample third-person character template contains an animated character mesh which is followed by a camera. That camera has its own settings and this includes field of view, or area of vision, which is the extent of the area that the camera captures. A wide field of view will let the viewer see more of the scene but this may scale down automatically if the display being used does not have enough resolution to support it.

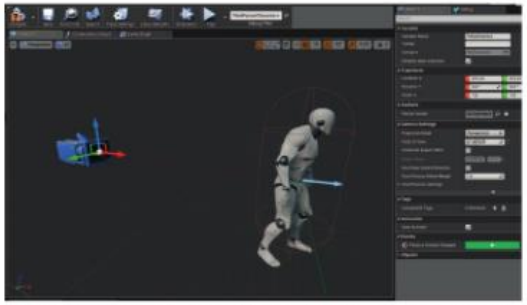

Figure 8.23: The field of view is set by the camera that follows the player's character in this third-person game

Link

For more about avatars and omnipresence see the Interaction model section.

Input methods

In Jump Chase, the player needs to be able to control the main character as easily as possible and there are a number of input methods available for this, depending on which controller the game will use. This game is played on PC, which means that it could be controlled by a keyboard/mouse or by a gamepad.

Figure 8.24: Player input can be mapped to keyboard, controllers or even both

The default settings for the third-person template in Unreal® Engine supports mouse and keyboard but it would be relatively easy to add gamepad support as well. While many different gamepads can be purchased for PC, the majority of them tend to have the same configuration so that they will work with the bulk of PC games. The code for the character will not specify a particular key or button but instead it will be named, for example MoveForward, Jump, LookUp. These labels are then mapped to one or more keys, so the walking movement may be mapped to the W, A, S and D keys. Some games allow menu options to provide customisable keys so that the player can choose exactly which keys they want to use. Some players may prefer the keys to be spread out, but some prefer them closer together. It can also work better for left-handed players to move the controls to the I, J, L and M keys. This feature is not usually available on console games, even though you can plug in a keyboard, but players are given control configuration choices in some games. Games consoles are sold with their own bespoke controllers and the designers will normally have used all of the button options available.

Asset integration

Once assets have been created for Jump Chase, they will need to be imported into the games engine. 3D modelling software exports meshes in a format called FBX that incorporates the mesh, textures and animations all in one file.

- Graphics:
  Graphics can be created in two different ways, using raster images or vector images. Raster images (known as **bitmaps**) are made up of a series of pixels and it is important when making games to keep the number of pixels in an image (its resolution) as low as possible without losing quality. This will make the game run more smoothly. A raster image with very few pixels will be a small file size but will have a very blocky style known as pixel-art style. The more pixels there are, the smoother the image. In Jump Chase the designers will need to make raster images for the textures applied to the 3D models and for the game's interface.

Key term

Bitmaps – images that are made up of individual pixels that together display a picture.

Vector graphics use points and lines to define an image. Vector graphics take more computational power to work with so are not used as frequently in games, and very few games engines support them. Even though vector graphics scale better, it takes less memory and processing to use a series of differently sized bitmaps.

- Texture mapping: All of the meshes that are imported into Jump Chase will have textures applied to them as set in the software in which they were created. One way to make a game run at a faster frame rate is to have different assets share the same texture file. This can be done when one asset, say, a floating platform, uses the top half of an image file for its metal texture and another asset uses the bottom half of the image. This means that two assets only need one image file. Textures are mapped to 3D meshes, which means that each flat surface on the mesh is tied to a particular part of an image file. If you think about a textured die, each face of the die would have a square in the image file and these squares would be mapped to each polygon in the mesh.
- Animation and video: The main character in Jump Chase is going to have different animation **cycles**. The default character has an **idle cycle**, a jump cycle, a walk cycle and a run cycle. The state machine uses the idle cycle as its **default setting** and, when the jump button is pressed, it will transition into the jump cycle.

Figure 8.25: Game characters can be rigged with bones to allow them to contain multiple animations

Key terms

Cycle – instructions or operations which are repeated.

Idle cycle – the animation that a character displays when the player is not moving them. A character in an idle cycle may stand still and breathe or tap their foot after a few seconds.

Default setting – the setting selected by the games engine when nothing else has been input by the developer or, if the game is running, by the player.

The walk and run cycles are blended together so that, when a speed value stored on the Jump Chase character changes (when the player pushes or releases one of the movement buttons), it will transition from idle to walk and then run, depending on how long the movement button has been pressed. All of these animations are used with the skeletal rig that is inside the mesh. Figure 8.25 shows what that rig looks like and how the games engine can be used to manually create new animations, if needed.

Other animations may be used in Jump Chase such as cut scenes or scripted events that move the story forward. Cut scenes will usually be recorded animations that are imported into the engine as a video file or, similar to scripted events, a series of animations controlled by the games engine. Unreal® Engine uses an animation system called Matinee that allows the developer to use existing assets and levels to create animated sequences, which are observed by cameras in the level. Therefore the player will lose control of the character while the cut scene or scripted event takes place.

- Audio: Audio is crucial to the experience of playing a game. The virtual world is made much richer by music and sound effects that create memories and by educating the player in how the world works. As for graphic assets, sounds can be imported into the games engine after they have been created in other software such as Logic®, Audacity® or GarageBand®. The level in Jump Chase will have fast-paced, frantic music that indicates that there is a time limit, but there will also be a constant ticking sound until the time runs out.

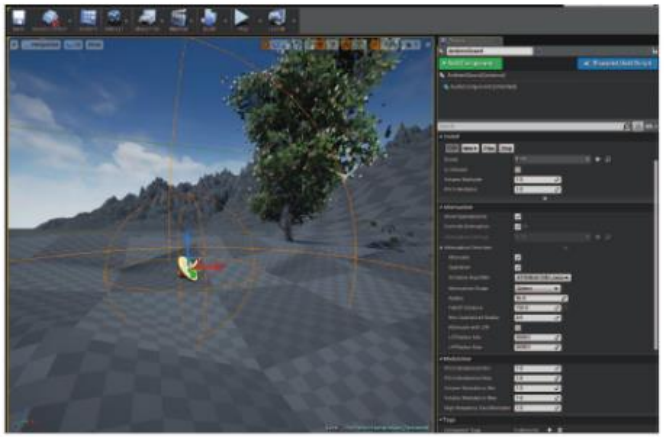

Figure 8.26: Audio can have 3D properties so that it can become louder the closer you get to an area

Sound assets can be 2D or 3D. 2D sound assets are heard by the player at its volume setting and cued to play as programmed by the developer. Sounds can use triggers, so when the player steps into a certain area, such as a creepy forest, they walk through an invisible trigger and a sound effect will play (the cry of an owl, for example).

3D sounds work in the same way but they are location specific, so as the player approaches the sound will get louder and as they move past it the sound volume will diminish. In Jump Chase, there will be a creepy building and the sound of someone laughing coming from inside a room that cannot be opened, just to add a sense of horror to the level and to keep the player motivated to move forward. The sound clips have to be synchronised to the visual displays so that they happen at the right time. This is especially important when sound is being used for spoken dialogue as the character mesh has to move its mouth at the same time.

Artificial intelligence

Now that Jump Chase has some assets in place, this is a good point to start prototyping some of the more advanced features to ensure that they work correctly. The game features some roaming robots that will move around the ground and make sure that if the player falls down then there is still a challenge as they try to get back onto the platforms above.

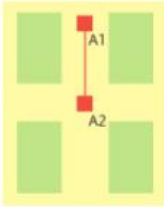

Figure 8.27: Grids are used in games to map levels for AI characters to traverse

The roaming robots will use a path-finding algorithm, a search algorithm, that needs to be tested in a prototype. The robots will consider the landscape to be a **grid** that is divided into separate **nodes**.

The code needs to be written such that the robots move between two different points. We need a way of working out the best way to move from one space to another and the breadth first search (BFS) algorithm can be used. BFS is an algorithm that shows the shortest route between objects on a grid. It does this by taking one step at a time and finding links between the different grid nodes. This is similar to a theory called six degrees of separation that says everything is six or fewer steps away, so that any two people can be connected, through a common experience such as being in the same film or going to the same school, by a maximum of six steps. BFS creates a queue of all of the nodes that are connected to the current node and then loops through them, one at a time, to see if they are connected or have been explored.

In C++, the programming language that the Unreal® Engine uses, the BFS algorithm would look like this.

```
//Loop until node queue is empty
while (!nodeQueue.isEmpty()) {
        //remove node ID from queue
        int visitedNode = nodeQueue.removeFromQueue();
        //check all connected nodes in a loop
        for (int newNode = 1; newNode <= n; ++newNode)
        {
          //check to see if the newNode is connected and if it has been
          visited
          if (isConnected(visitedNode , newNode) ) {
                  if(!hasBeenVisited[newNode ]){
                          //add to queue
                          nodeQueue.addToQueueEnd(newNode);
                          markAsVisited(newNode);
                  }
          }
        }
}
```

Key terms

Grid – a network of lines that is projected onto a game level in order to split it into logical sections.

Node – a point where lines intersect.

Link

For more about path-finding search algorithms, see the section Search algorithms.

This code snippet shows a section of the BFS algorithm where it is checking through all of the nodes connected to the current node and seeing if they have been visited yet. The robots in Jump Chase will do this before they move. Once the code loop has finished and created a list of nodes to move through, the robot will then make its moves.

More advanced features

- 3D rendering: Jump Chase has a 3D environment with sculpted landscape and audio. This is created in Unreal® Engine and the platforms will be 3D meshes created in Autodesk® 3D Studio Max.

- Save game states and player progression: Another advanced feature is creating save files or auto-save points. Many modern games auto-save all the time, especially online titles. Third-person platform games like Jump Chase tend to use checkpoints, parts of the level that once reached are automatically returned to if the player's character loses a life. A checkpoint location will be stored in a save file locally on the player's system or on a server if it is an online game.

  Local file saves are also used to save progression information in Jump Chase. Details about how far the players have reached and how quickly they completed the challenges are added to leaderboards. The game will also use an achievement system so that when a player achieves certain goals in the game, such as completing it on the hardest level of difficulty, they will be able to see a trophy on the achievement screen.
- Multiple players: Games developers have to think very carefully about any multiplayer or networked features they are going to put into a game. Multiplayer games contain many advanced features such as player matching, which is a system that finds different people around the world who are of a similar level to you and will provide an appropriate challenge for you. Players do not like to be pitted against opponents who are too strong or too weak for them as this makes the experience unsatisfying.

  Ensuring that network connections are maintained is important, but this is not something that is in the developer's control. Game publishers will be responsible for creating servers that online games connect to, and the reliability and availability of these servers is dependent on how much money is spent on them. The servers have an important function in allowing the player's systems to communicate with each other and in maintaining any online leaderboards and achievements.

This example is designed to give an overview of a developed game but is too complicated to be created by a single learner or a small team. The scope of this unit is to provide an understanding of the whole process and to create a simple working game that will provide an understanding of how to create something that could be developed into a wider, working title.

Testing computer games

Once the development is completed to a point where testing can begin, it is important to get started as the more time that can be given to testing, the better the final product will be. The focus of the testing is split into four areas that look to answer the following questions.

- Playability – do all the features in the game work? Can the player do everything that they should be able to do? Is it possible for the player to get stuck anywhere?
- Compatibility – does the game work on its intended platform? Does it load and save appropriately? Do all the system's controllers work with the game?
- Stability – can the game be played from start to finish without breaking? Is the frame rate consistent throughout?
- Acceptance – do the players enjoy the game? Is the challenge at the right level?

Testing tools

Games engines and IDEs contain tools that help with the testing phase of games development. Figure 8.28 shows a statistics window that can show details of all of the assets in the scene and how they are being rendered.

Figure 8.28: A statistics window

Key term

Bug – a problem in code that causes errors or glitches in a game.

Other tools include debugging tools like breakpoints and watch values that allow the programmers to automatically stop the game at certain lines of code within a function to check to see which values are being stored. This will be useful if a particular game feature is not working correctly because it will help the programmer or tester to solve the problem (or **bug**) by processes of elimination. When the programmer knows where the game is going wrong, they can figure out how to fix it.

Feedback

Feedback from testers, clients and players is very important and this is why end user testing or user acceptance testing is crucial. Testing by players may be done by small focus groups at the development studio, in larger sessions at games conventions or a game may be opened up to the public as part of an open beta. It is common for small testing groups to have to sign non-disclosure agreements (NDAs), which are contracts that prevent them from revealing details about the game. This way, if anything is taken out of the final version of a game, the public will never know about it.

End user testers will be asked to consider the following.

- Effectiveness – how well does the game play? How good are the controls?
- Presentation – does the game look good? Does the art style work well?
- Performance – does the game run smoothly? Has it lagged at any points?
- Accessibility – how easy is the game to control? Is the difficulty level manageable?
- Portability – does the game work well on different platforms? Does it work well on different screen sizes?
- Robustness – can you break the game? Can you get stuck anywhere?
- Purpose – do you understand what you are meant to be doing in the game? Do you understand why you are doing it?

Refinements

The feedback gained from users will go into making refinements to the game, but the scale of refinement depends on how much time is left before the final deadline and may also be dependent on the budget. Small changes to layout and challenge level can be considered but anything that requires new meshes or characters will be impossible

after the beta stage of development. Some console games developers, when criticised on the performance of their games, have blamed the differences between console development kits and the retail versions of the consoles that players have at home. The disparity can be because the retail versions are updated more frequently than the console development kits in order to solve security issues. But if the console development kits do not have these updates, then developers are essentially having to test their games on slightly different systems which can lead to problems.

Theory into practice

Skills, knowledge and behaviours

If you want to work in the computer games industry, you need to develop the following skills.

Planning and recording

- You need to be prepared for the massive workload that is involved in the games development process. You must have skills that include the ability to set your own targets, consider timescales and decide how you are going to get feedback from colleagues.

Reviewing your own work

- You must be adept at reviewing your own work and be willing to admit when something is not as good as it should be. Teamwork is not the place for ego and if you cannot put your hands up and ask for help when something is not going to plan, then working in creative teams such as games development studios is not the right place for you.

Responding to feedback from others

- You must be able to collect useful feedback from other creative professionals and end users, which means putting your product in the hands of critics and being able to listen to both positive and negative feedback.
- You must be able to look at the feedback that you receive and compare it against the original requirements of the brief. If an end user is demanding unrealistic functionality, then their feedback is not that useful, but if someone is in your target audience and they are not enjoying the game, you must find out why and try to see if you can do something to improve their experience.

Reviewing computer games

After a game has been tested, it will then be reviewed. The review of a game is usually completed by the development team and the games publisher. It could be tempting to skip the review stage if deadlines are pressing but it is a really important step as it is an opportunity to evaluate the entire design and development process.

Quality

The reviewer will first comment on the quality of the game. How well is it made? They will consider the textures, the animations, the game mechanics, everything. Players tend to pay around £40 for a console or PC game or half that for a digital download game for PC. They expect to get a quality product for their money. This means that all the controls need to work well with the game, the graphics need to look good and the challenge levels need to be appropriate for the intended audience. All of these aspects will be checked by the reviewer.

Suitability for audience and purpose

The reviewer will consider who the audience is for the new game. They will make comments about whether or not the game has been designed with the audience in mind and to what extent the audience will be satisfied by the title. It is not just a question of age here – it could be a focus on the ability of the players, their intelligence or their history with that franchise of games. If a sequel comes out that does not refer back to the original story or move the story forward, then it is not considering its audience, it is just trying to make more money without giving the audience what it wants and expects from a sequel. The game must also be reviewed against the original purpose intended for the game.

Original requirements

The reviewer will also look at the original requirements of the game. They will look back to the original high-concept design document and comment on whether or not the game has met the outcomes that it originally set out to achieve. It may be that changes and refinements during development have changed some things, but a game should generally end up in a similar state to how it was originally conceived and designed.

Legal and ethical constraints

The games publishers should get a legal team to review the game and its content if there is any chance that it breaches any legal constraints such as copyright law or ethical considerations. Smaller development studios will not have the luxury of their own legal team and may have to check potential legal issues themselves or pay for legal advice from an external firm.

Technology constraints

Once the game has been made, its technical requirements will be confirmed and the reviewers will be able to check to see if these cause any problems. In the previous generation of consoles, the Xbox 360® used DVD discs and if a game was too big it would spill onto a second disc, which meant more expensive boxes were needed and players would be frustrated at having to change game disc halfway through the game. PC gamers pay very close attention to the recommended technical specifications so that they can play their games without having to sacrifice graphics or quality. When the quality team reviews the game, they will set out the minimum technical requirements for the game and the recommended requirements, as they will be advertised.

Strengths and improvements

A game is always sold on its strengths, and the reviewers will make sure that they are fully aware of everything that the game does well and anything that is not perfect. If a game is found to have an issue and it is too late to make fixes or refinements, then the developers must produce a **patch**. Games which have been shipped by the time problems are found will have what is called a 'Day 1' patch. A 'Day 1' will automatically download and install the fix when the player goes to play the game for the first time, but only if the player's system has internet access and the game knows to check for patches whenever it loads. Patches can continue to be delivered to the player's system long after the game has been released, but the longer the delay between fixes, the more dissatisfied the player will be.

Key term

Patch – a series of code fixes that are downloaded and applied to the game code in order to fix problems.

Platforms and compatibility

The review team will look at all the different platforms that a game has been developed for and ensure that it works correctly and runs effectively on those platforms. There may be a certification testing phase where game code is sent to console manufacturers or digital distribution platforms to ensure that the game's install process works properly and that it makes proper use of any system features that it needs, such as network access or game saves.

External reviews and quality characteristics

Video games are big business, with millions of pounds a year being spent on games development and even more being spent on research into new hardware systems and advances in technology. Games journalism is crucial to both players and developers because the review scores that games are given before they are released can make or break it. While the games magazine industry is starting to fade, games websites are more popular than ever and players will often look to IGN.com, Kotaku.co.uk and Gamespot.com to decide what their next purchase is going to be.

Some games reviewers have got into trouble in the past for giving positive reviews to highly flawed games and have been accused of being influenced by games publishers through being given trips and lavish gifts. This has led to an increase in popularity of Metacritic.com, a site that gives a single score for a game based on the average of the scores from many different review websites. It also allows users to give a score to games, so that the opinion of the audience, as well as that of the critics, can be seen.

Further reading and resources

Game software to download for free

Unreal® Engine 4 – www.epicgames.com (free for personal and educational use).

Unity® 5 – www.unity3d.com (free for personal use, educational fees vary).

Autodesk® 3D Studio Max – www.autodesk.com/education (3 year licences for free for students and educators).

Tutorials

Brackeys.com – brackeys.com excellent free tutorials.

Pluralsight Creative – www.pluralsight.com subscription based video tutorials.

Articles

Richard Bartle – 'Hearts, Clubs, Diamonds, Spades: Players Who Suit MUDs'. http://mud.co.uk/richard/hcds.htm.

David Keirsey – 'Please Understand Me II' (1998); www.keirsey.com/.

Assessment practice 8.2

B.P3 B.P4 B.M2 C.P5 C.P6 C.P7 C.M3 BC.D2 BC.D3

Now that you have shown your knowledge to the games publisher about different games platforms, you have been asked to design a game. The publisher wants you to create a game for the web using the recent WebGL plugins that are available on the main games engines.

Remember that WebGL allows developers to create whole 2D and 3D games that will run in a web browser

In your design, you must show that you understand the constraints of designing for a web platform and that nothing is asking too much of the system.

The game's design is your choice. The publishers are open to seeing your ideas but they have specified the following requirements:

- a PEGI rating of 7 or lower
- either a 2D or a 3D design
- a third-person player character.

You should produce a GDD document and a set of design documents that justify your design decisions, showing how they fulfil the purpose of your game and meet the client's requirements. You will need to review the designs with others to make refinements.

You might create the following:

- storyboards
- level designs
- a summary of game mechanics
- game screen mock ups.

Having completed the design for your WebGL game, it is now time to create it. You have been asked to produce two working levels for the game that you have designed with a welcome menu and a game over screen.

Remember that you have learnt how long and complicated the game development process can be. Do not be too ambitious in your design. You can achieve high grades with a simple game design that works well.

You must ensure that the game is optimised for web delivery, which means that textures should be minimal resolution, any 3D meshes should be low poly and any audio should be kept to a minimum. Keep a production diary of your development and testing which records your organisation and self-management.

After you have created the game levels, you must test it fully and produce a review for the clients that evaluates the design.

Plan

- What am I being asked to do?
- What sort of game would the audience want?
- What resources do I need to complete the task?

Do

- Have a got a full set of design documents?
- Have I thought about everything that needs to be designed?
- I can identify when I have gone wrong in my game production and get myself back on course.
- I am recording my own observations and thoughts in a production diary.

Review

- I can consider all constraints involved in a web game.
- I realise where I might not fully appreciate how many assets I need to create to make the game look professional.
- I can explain the skills that I employed and the new ones which I have developed.
- I can explain what success looks like.