Proof-of-concept 3D level creation tool for blind gamers

Proof-of-concept 3D Level Creation Tool for Blind Gamers*

Matthew Tylee Atkinson
Loughborough University
M.T.Atkinson@lboro.ac.uk

Colin H. C. Machin
Loughborough University
C.H.C.Machin@lboro.ac.uk

Presented at CSUN 2009

Abstract

We present a prototype tool that allows blind gamers to create 3D levels for a mainstream first-person action game for the sighted. The system is designed to abstract many of the details of level design so that, for example, aesthetics or precise coordinates of objects need not be specified by the user. Though the system has been created with accessibility as the primary goal, it also brings about the possibility for the computer to dynamically generate game environments.

Keywords: Accessible Games, Audio Games, Multimodal Interfaces, Level Design

1 Introduction

Mainstream computer games—particularly 3D action games—are both very popular and have large online communities surrounding them, providing means for social interaction ranging from competitive and collaborative play to news websites and discussion forums (for examples see PlanetUnreal[1] and PlanetHalf-Life[2]). Further, many games are now released along with some of the tools used to create them, often including level editors and facilities to allow gamers to modify the behaviour of the game.[3] This provides amateur developers with the ability to program new game-modes, weapons or enemies and distribute them to the rest of the community. An even more popular activity is that of creating new levels, often referred to as *maps*, for games—almost every major game has a mapping community and there are far more maps than programmed modifications. An inherently visual CAD-like program such as the one shown in figure 1 is used to create maps.

Due to the often visual and fast-paced nature of modern games, few are inherently accessible to people with disabilities. The goal of this work was to build on prior work that provided access to mainstream games and their development tools, to allow blind gamers to create their own levels for 3D games.

1.1 Accessible and Audio Games

Accessible games are computer games that have been designed to be (or simply are) playable by people with disabilities. The challenges involved in making games that are accessible are significant, covering the appropriate presentation, prioritisation and filtering of information, often in a time-critical manner.

It is important to consider the accessibility of games for reasons of social inclusion, emphasised by the fact that the potential educational qualities of games are being considered[4] [10, 5]. Many popular user interfaces are gradually incorporating similar technologies and paradigms as are employed in games[5] [13]. Game-like paradigms are being applied to collaborative and social environments[6] and it is vital that access technologies keep up *and are adopted* so that disabled people are not excluded as these changes take place [12].

Many accessible games are created specifically for disabled people, often by disabled people, and span many styles including action[7] and arcade[8]. Others, such as "audio games" [3] are accessible to those with vision impairments by virtue of the fact that their primary means of output is sound. A number of research projects have sought to make games that are accessible to those with other disabilities, such as

[1] http://www.planetunreal.gamespy.com/ (all web pages last visited on 11/03/2009)

[2] http://www.planethalflife.gamespy.com/

[3] UnrealEd ships with games using the Unreal Engine http://udn.epicgames.com/Two/IntroToUnrealEd.html and Valve's SDK is available free of charge http://en.wikipedia.org/wiki/Source_sdk.

[4] FutureLab: Teaching With Games http://www.futurelab.org.uk/projects/teaching_with_games

[5] http://en.wikipedia.org/wiki/Expose_clone

[6] http://www.secondlife.com/

[7] Shades of Doom http://www.gmagames.com/sod.html

[8] Alien Outback http://www.draconisentertainment.com/products/?id=ao

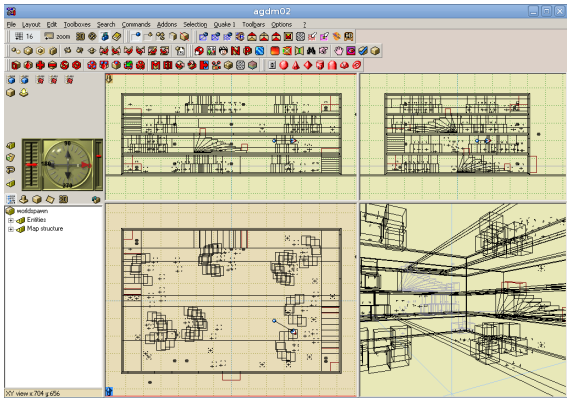

Figure 1: A Representative Graphical Level Editor ("QuArK").

motor-control[9] difficulties. There has been growing interest in making games that are playable by people both with and without disabilities simultaneously [14] and some organisations have lobbied the mainstream games industry to improve the accessibility of their games [7, 6].

There have been some examples of successful attempts to incorporate accessibility features into mainstream games, such as closed-captioning[10]—an example of where mainstream developers *have* included accessibility features [2, section 4.1]. The AGRIP project has provided access to a mainstream 3D game for the blind [1].

2 The Problem Domain

This section discusses current practices and outlines the contributions made by our prototype solution—Level Description Language (LDL).

2.1 Mainstream 3D Level Editing

The standard procedure for creating levels (often referred to as *maps*) for games using the Quake engine, from id Software, is as follows. The basic process is the same for most other current 3D game engines, though the specific file formats and tools differ.

1. The user employs a graphical editor, such as that shown in figure 1, to define the structure and content (lighting, game-specific items) of the level. This works in much the same way as a CAD system. In the case of Quake-engine games, *constructive solid geometry* is used—i.e. the world begins as empty space, into which solid shapes must be placed to form the structure of the level.[11] A well-formed level must have no gaps in its structure; these are known as *leaks* and can cause rendering anomalies.

2. The level is then exported to the standard format for the particular game in question—a `.map` file in this case. The editing application may have its own custom file-format, so that higher-level primitive shapes (such as staircases and hollowed rooms, as opposed to just solid boxes and other primitive shapes) can be more easily constructed, but all editors are able to export `.map` files to ensure compatibility with compilation tools.

3. Finally, the `.map` file is compiled into a `.bsp` file for use in the game. This is usually achieved in three steps by the standard compilation tools: **bsp**, which compiles the human-readable `.map` file into an efficient binary structure; **light**, which calculates light levels in the map and **vis**, which calculates which parts of the map the engine can avoid having to draw given the player's viewpoint, thus increasing efficiency.

2.2 Accessible Tools for Level-Editing

For blind and vision-impaired people, the "Non-Visual Auditory Authoring" approach to audio game design [9, section 4] would be both accessible and somewhat game-like in nature. The "Audio Game Maker"[12] is a tool for the creation of entire audio games and presents an audio-based interface.

Though the consideration of disabilities other than vision impairments is out of the scope of this paper, it should be borne in mind. In the case of making maps for Quake, deaf people would be able to use the standard interface of most editors. However, some motor-impaired users would likely find it very difficult to create levels in this manner, as it is very mouse-intensive. The fact that different interfaces will be needed by different disability groups implies a need for a common high-level *map description format* to be developed before any user-specific editing applications should be created.

The accessible editing tools that currently exist tend to put many constraints on what can be created due to either the nature of the specific accessible game or the difficulty in creating a flexible editing environment. For example, the editor for the arcade-style action game "DynaMan," included with the game, only permits levels to be laid out on a grid. This is

[9] `http://www.oneswitch.org.uk/`
[10] `http://gamescc.rbkdesign.com/`
[11] Under this system, a room is composed of six solid blocks.
[12] `http://www.audiogamemaker.com/`

perfectly adequate for a fast-paced arcade game, but it was decided that for the case of 3D layout, this approach would not be scalable.

Most current "3D" accessible games are not fully 3D. As with early first-person mainstream computer games, the accessible games are actually a series of 2D environments[13] that are linked together to give the user the impression they are in a larger 3D space (this is achieved by the use of lifts between floors in "Shades of Doom", for example).

2.3 Motivations and Contribution

As discussed above, the current problems of accessible game level editing are that: (a) there are not many editors; (b) there are no true 3D accessible level editors and (c) there are no editors that allow disabled people to make levels for mainstream games (because most mainstream games are not accessible).

The goals of the AGRIP project are to provide access to a mainstream game, online community, development tools and level editing facilities in order to promote inclusion of disabled people. The latter of these goals is the focus of this paper.

Not least of the reasons for investigating accessible level creation tools is that a survey carried out previously by the authors indicated the demand within the AGRIP and Accessible Gaming communities [1, section 4]. Out of 20 people that took part, 18 said they definitely wished to make maps and the remaining 2 said they may wish to do so. It is also conceivable that the system developed here could be of use to non-disabled people who simply prefer a different method of working than the CAD-like systems described above.

The work described here makes use of AudioQuake as the delivery mechanism for the maps generated. Though this game is based on the original and now somewhat-antiquated Quake game engine, the techniques discussed here are applicable to later engines.

3 Level Description Language

This section describes the requirements, design, pertinent implementation details and justification of design decisions for the proof-of-concept system developed.[14]

[13] e.g. connected areas/rooms but with no variation in height

[14] All code and documentation for the system may be obtained from `http://www.agrip.org.uk/ldl/`

3.1 Requirements

In the previous section, the key requirements for the system were identified, as follows.

Compatibility with existing game engines, standards and tools—i.e. allow disabled people access to *the same* systems that other gamers use.

Accessibility of both the editor and output levels—to the blind and vision-impaired in the first instance.

Layered design to allow alternative front-ends (user interfaces) and back-ends (game engine formats) to be supported. This allows the possibility of granting access to those with other disabilities later, as well as interoperability with other engines and editing tools.

Automatic production of aesthetically-pleasing maps that a sighted person would enjoy playing immediately or, at least, that could be modified using a traditional editor such that a sighted person would enjoy playing them.

The prototype system addresses all of these requirements. A pilot survey carried out with members of the AGRIP community (described in section 4) indicates to what extent the requirements were met.

3.1.1 Scope

The focus of the present work is on making the description of 3D space accessible and assessing how this can be improved through the development of other interfaces. Game-specific and minor features such as allowing the full range of power-ups to be placed in a level are out of scope of this paper. The system developed was designed to allow the user to produce deathmatch[15] maps. Future development will focus on improving game play, particularly for single-player levels, and aesthetics (to improve appeal to sighted gamers).

In this initial work, the available geometric shapes are rather primitive in comparison to those that the Quake engine can support. To begin by implementing very complex architecture would (a) have taken considerable time and (b) potentially caused the system to be impractically difficult to use bearing in mind the relative simplicity of current accessible editing systems and architecture in Interactive Fiction (IF)[16]

[15] a type of online game where each player must eliminate the other players to earn points

[16] Text-based story/adventure games such as: Spider and Web; The Lurking Horror and Zork.

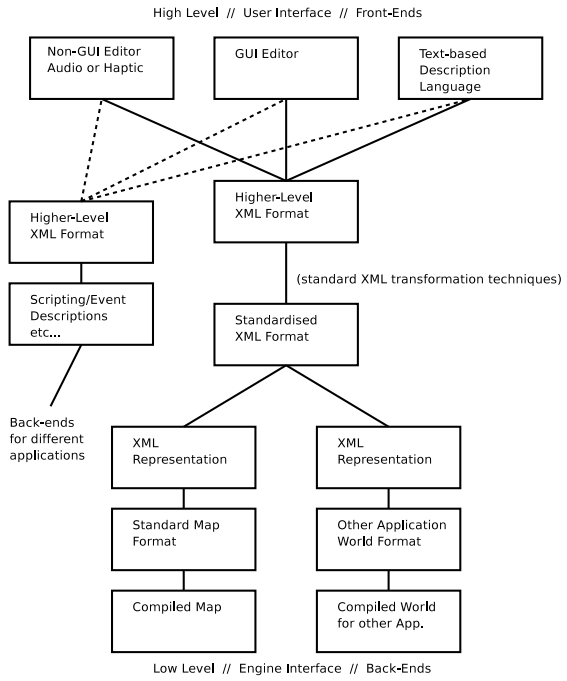

Figure 2: Previously-proposed architecture for an adaptable editing system for 3D applications and games [1, figure 2].

games (which is what the target audience are familiar with). This is discussed in more detail in section 4.2.2.

3.2 Layered Architecture

A potential architecture for a level creation system, where high-level input from the user—essentially a description of the level—is transformed into the standard `.map` format described above was proposed previously [1, section 3]; see figure 2. As discussed in the earlier paper, a layered approach may afford many benefits, including the ability to develop multiple front-end tools and the possibility of inserting extra stages into the processing pipeline to carry out adaptations to the level for people with particular needs or tastes.

The system was implemented as a sequence of Python scripts. The top-level XML file, written by the user, is piped into the first of these scripts, which outputs an XML file at the next lowest level. The scripts are executed in a pipeline to transform high-level descriptions step-by-step into `.map` files,[17] which are then compiled using the standard procedure. They may also be opened and refined in a traditional graphical level editor.

3.3 Designing for Non-visual Description

The layered approach described below was used for more than the technical reasons outlined above: it affords the user some choice over how much detail they are prepared or able to give to the description system. It also allows for details such as aesthetics to be abstracted (see section 3.4.3) and can promote co-operation with sighted level-designers due to the interoperability of formats used.

One key aspect of the system is the "view" of the map that the author is required to hold during the description stage. A top-down point-of-view onto the map was used, along with compass directions (plus "up" and "down") for specifying spatial relationships. This was largely for reasons of familiarity—many of our target users are familiar with this representation from Interactive Fiction and it seemed the closest match to the way that people navigate in Audio-Quake and other accessible games. The user survey (described in section 4) sought feedback to assess the validity of these assumptions.

Another issue is that of the map's overall layout in terms of the routes between different parts of it. Often for blind and vision-impaired people, getting an overview of a particular situation (equation, picture, place, game level) is very difficult, as they often do not have the sensory capability to extract an overview. This could make using a system like LDL extremely difficult due to the layout approach described shortly. The pilot survey also sought opinions on how hard this task was and how it may be improved.

It was not clear from the motivating survey discussed above—and subsequent discussion with potential users—what type of interface would be best suited to level description. For this reason we decided to develop a high-level XML dialect and elicit feedback from users as to what sort of interface(s) they would prefer in future.

3.4 Layer Descriptions

Here we discuss each layer of the description system in turn.

3.4.1 3D Positioning of Rooms

The highest level in the system is that in which rooms are linked to each other in 3D space. Input is given by the user in the form of a simple XML dialect. The

[17] A limited subset of the reverse process—converting `.map` files upwards—was also developed.

Listing 1: Most basic usable map in LDL.

```
<map name='Hello, World!' style='base'>
  <room id='start'>
    <item type='info_player_start' pos='c'/>
  </room>
</map>
```

Listing 2: A very simple deathmatch level.

```
<map name='Very Simple DM Arena' style='base'>
 <room id='main' size='big'>
  <con wall='s' target='spawn_a' type='door'/>
  <con wall='n' target='spawn_b' type='door'/>
  <item pos='c' type='item_armorInv'/>
  <item pos='e' type='weapon_rocketlauncher'/>
  <item pos='w' type='weapon_grenadelauncher'/>
 </room>
 <room id='spawn_a'>
  <item pos='c' type='info_player_start'/>
  <item pos='c' type='info_player_deathmatch'/>
  <item pos='n' type='item_rockets'/>
 </room>
 <room id='spawn_b'>
  <item pos='c' type='info_player_deathmatch'/>
  <item pos='s' type='item_rockets'/>
 </room>
</map>
```

most basic usable map is given in listing 1 and a small example map is given in listing 2.

These examples demonstrate the basic structure of a map—a series of connected rooms that can contain items (such as weapons, power-ups and player start points). The map in listing 1 is a single, medium-sized (the default) room that contains only a player start point, positioned in the centre of the room. The map in listing 2 consists of a larger central room that connects to 2 medium-sized rooms that house the player start points. Figure 3 shows part of a larger LDL map. Other typical game items are distributed about the map (by means of compass points). The following are the key design features for this layer.

Order is unimportant when specifying rooms, items and connections. It was anticipated that most users will specify the rooms and connections as a series of routes from the start through various parts of the map. This is discussed further in section 4.2.2.

The positioning of items within rooms can be achieved using compass points, as if one is looking down on the room. An optional height can be specified—e.g. "nw 20%" signifies that the item should be placed in the north-west of the room, when looking down on it from above, and at 20% of the room's height above its floor.

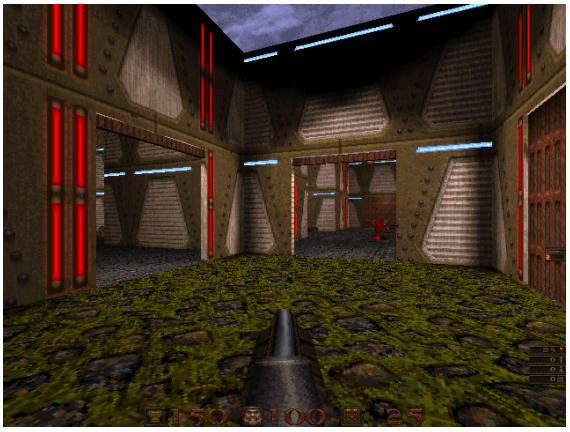

Figure 3: Another example map.

Other coordinate systems may be used, such as an offset from the room's origin[18] as a fraction of its width, depth and height. For example: "50% 25% 50%" is a point half way across the room, a quarter of the way from back to front and half of its height above the floor. This is actually the same as the compass point "s" (south) in the current implementation.

A connection between rooms is positioned on a given wall of the rooms involved, indicated by compass directions. A connection on the north wall of one room will require a counterpart connection to be made on the south wall of the target room—two connections are needed so that holes are made in the appropriate places in the walls of *both* rooms, thus allowing the player to pass between the rooms. It is not necessary for the user to explicitly write in the reverse connection (e.g. in listing 2 only the connections from the central room to the other rooms were specified).

Rooms are currently limited to being cuboidal in shape for reasons of both ease-of-use and implementation simplicity, however their internal shape may be modified by the introduction of other solid bodies and contained rooms.

The location in 3D space of each room is determined by the connections between it and other rooms. The first room encountered in the XML file by the script is given a default origin point. Any rooms connected to this room will have their origins calculated relative to the first room, via the connections. All rooms must be

[18] the corner of the room with the smallest x, y and z coordinates—its bottom left-hand corner as we look down on the map

```
<map name='Advanced Connections' style='base'>
  <room id='start'>
    <item pos='c' type='info_player_start'/>
    <con type='door' target='other' wall='n'
      pos='t' elevtype='stairs'/>
  </room>
  <room id='other'/>
</map>
```

The connection is positioned at the top of the wall on the "start" side, causing the origin of "other" to be higher and the elevation device (stairs in this case) to be built.

Figure 4: Example of simple 3D layout: LDL code.

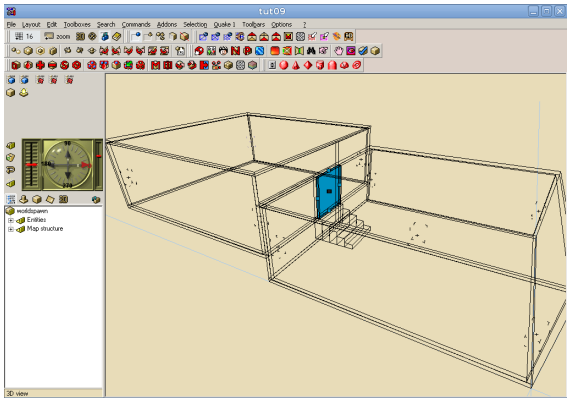

Figure 5: Example of simple 3D layout: overview of the layout in QuArK.

connected to other rooms and rooms may be nested inside each other.

True 3D layout is achieved by having rooms at different heights. This is made possible by the notion that connecting doors/holes in walls may be positioned at any point *on the wall face* (as if one is standing in front of the wall looking at it). By adding an extra `pos` attribute to the <con> element, this can be achieved. An example is given in figures 4–6, where the `pos` attribute has been set to "t" (top)[19].

As this is (currently) the highest level in the system, any alternative user interfaces to the system would be expected to output XML in this format. Compliance to the format may be easily ascertained by making use of an XML schema.

[19]Here the positions "top", "bottom", "left" and "right" (and combinations) are used to distinguish the activity of placing a connection on a wall from that of laying out the map as a whole. In fact, any coordinate system supported by LDL can be used in any place, but it was felt that this could easily confuse matters for new users.

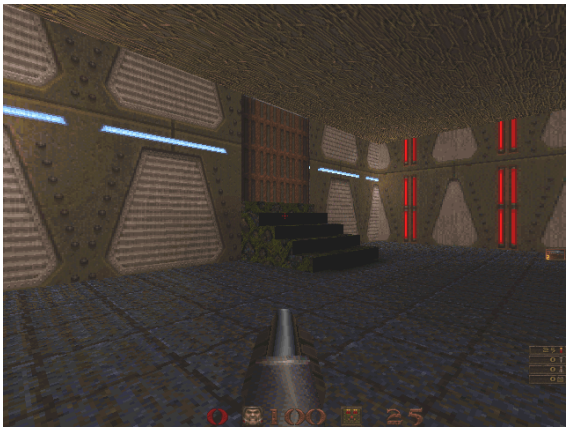

Figure 6: Example of simple 3D layout: the level in-game.

A possible higher-level layer would allow the user to simply specify the rooms and which rooms are connected and leave the routing to the system. This is out of the current system's scope, however (but the possibility is discussed in section 4.2.2). Other work has proposed a method for interacting with a generated 3D scene with one's voice [4]. Though it uses visual feedback to allow the user to verify the scene manipulations, it could be adapted for use in an accessible system such as LDL.

3.4.2 Builder Macros

In the top layer, rooms and connections can be specified to have a number of complex structural features within them, such as doors, stairs and elevation platforms. In traditional editors, these features have to be manually created from simpler building blocks and some—interactive elements such as doors or switches—have to be flagged as *entities* to the engine, so that the game knows how these pieces of the structure should behave in-game (i.e. the engine uses the entity type to attach some executable code to that particular object).

This stage in the chain builds these sometimes-complex entities within the bounding area specified in the top layer, and applies any relevant properties (such as the direction of slope of stairs) passed down. This allows the user to create complex structures easily—they only need to specify the important parameters (e.g. step height or if a key is required to open a door). A similar approach, known as "prefabs", has been taken for complex structures has been taken in later 3D action games, so that level designers and modellers can work separately.

3.4.3 Lighting and Textures

A number of pre-defined lighting sources and textures for use in levels are provided by the game engine. Clearly it would be difficult or impossible for a blind person to apply an aesthetically-pleasing lighting scheme to a level, such that a sighted person would either be able to enjoy playing it (or at least need to modify it only slightly to make it enjoyable). This stage of the process compensates for this by selecting a lighting style from a pre-defined list based, on the `style` attribute of the map and the size of the room being lit.

The styles are defined in a separate XML file and allow for elastic grid-based lighting that guarantees a suitable minimum separation of lights. Separate grids can be set up for perimeter and central lighting, as well as different offset coordinates in each dimension (so, for example, central lights can be placed higher than those attached to walls) and different light object models rendered by the game.

Another aspect defined in the style XML file is that of textures. The `style` attribute of the map determines the texture set to be used. This attribute may also be specified for any room individually and is inherited by all of its children. In summary, the style file contains the following.

Lighting Styles as described above.

A list of textures that maps friendly texture names and descriptions to the often-cryptic texture names used in the game.

A list of texture sets to be applied to the different surfaces of a room. A texture set is a "known-good" set of appropriate textures for the room and structural objects inside it. The technique is used so that blind users know they can select a sensible appearance.

A textual description of the style, texture or set of textures that attempts to impart the "feel" of the particular style in question (e.g. military base, or medieval).

Figure 7 shows an example of the flexibility of the styles system. The map from figure 6 is rendered in "base" style. After changing the style to "medieval" the map changes significantly.

To improve the aesthetics of maps in future, it is proposed that randomness be introduced to ensure that not all rooms and lighting styles look too similar. Finally: the adoption of high-contrast styles (comprised of custom textures) could perhaps enable vision-impaired users to navigate maps more easily, without having to use an entirely audio-based rendering of the game world.

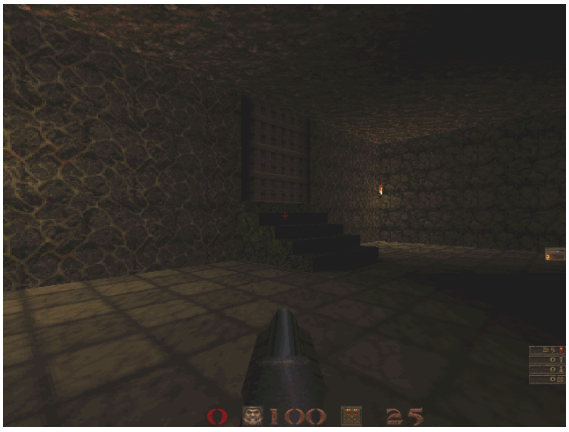

Figure 7: Map from figure 6 in "medieval" style as opposed to "base" style.

3.4.4 Rooms → Multiple Separate Solids

As discussed above, Quake engines use constructive solid geometry. One effect of this is that there can be no concave solids (*brushes*) in the map. A hollow cuboidal room must therefore be constructed out of 6 cuboidal solids that form the sides. To prevent leaks, there must be no gaps in the map, so any room that is not inside another must be totally sealed.

This stage breaks rooms (and connecting corridors—any hollow area specified above) apart into solids and puts connections at the right places by building walls around them (and inserting a door in the gap if need be). The editor "QuArK" has a similar feature, designed to make easier the moving around of doors and connected rooms, by having the editor divide walls up to create holes in the correct places.[20] This stage guarantees that LDL-generated levels will have no leaks, which can be useful for all users as opposed to just vision-impaired people.

In future work, structural styling artifacts (such as turrets for medieval-style levels) may be implemented at this stage.

3.4.5 The `.map` File and XML Analogue

A `.map` file is a text file that, as well as listing the entities (such as lights and power-ups) in a level and their properties, describes each of the solid brushes in a map as a series of intersecting planes. The last two stages in LDL break the brushes down into planes

[20] . . . therefore it is at this stage that conversion to the QuArK `.qkm` format could take place, which would allow changes to be more easily made by QuArK users.

and convert this, expressed initially in XML form, to plain text—in the `.map` file format.

4 Results—User Survey

To test the assumptions discussed above and to ascertain the potential of the system, a survey was carried out with members of the AGRIP[21] mailing list. The reason for targeting such a small subset of the entire accessible gaming community was that to accurately test LDL, the users needed to have familiarity with AudioQuake. It was deemed that more informed and reliable answers would be obtained from those on the mailing list, as they would have the requisite experience with the game to effectively test LDL.

4.1 Questions and Rationale

The full questions and answers data can be obtained from the LDL website given in the introduction above. The survey was broken down into 5 main sets of questions, as follows.

The user's vision impairment. Participants were asked about their impairment and spatial awareness, with the goal of ascertaining how useful LDL may be to people with different visual and spatial capabilities.

The user's experience with other accessible and Interactive Fiction games, as well as markup languages (HTML and XML). The intention was that these details may help us determine why a user may have problems with the system.

The navigational capabilities of the user (i.e. their mobility skills) and if they felt that there were any similarities between navigation in the real world and AudioQuake and LDL.

Usage of LDL including the factors that the user attributed errors to (including the strictness of XML, their spatial awareness, difficulty in conceptualising a whole map's layout and LDL not providing the required features). This section also asked if the user gave up with the system (and why) and if the results were what they expected.

The participants' opinions of LDL were also collected—i.e. if higher-level layout features are required; what sorts of interfaces they would like to the XML format; if they would like any extra features and, finally, how novel they felt LDL was, how it compared to other similar tools that they have used and how much potential they thought LDL had to be improved.

[21] `http://www.agrip.org.uk/`

Table 1: User information.

Property	Modal Answer	Num
Programmed before?	Yes	5
Written HTML?	Yes	5
Written XML?	Yes	4
Played 3D games?	Yes	3
Played IF?	Yes	5
Desire for LDL	Total	3
Mobility Skill	Excellent	3
AQ Relevance	Strongly, Totally	2, 2
LDL Relevance	Moderately	4

The survey was answered in full by 7 members of the AGRIP community; certainly not enough for a detailed statistical analysis but likewise definitely enough to enable us to determine if the current prototype is worth developing further—and in which direction. Naturally one very powerful way to asses if a system actually works is to collect the output users create with it. A collection of user-made maps can be found on the LDL web page.

4.2 Findings

The findings of the survey indicate that the system should be further developed and suggest a number of further research avenues that should be pursued.

4.2.1 User Backgrounds and Mobility

Information about the users is given in table 1. All users who took part in the survey were blind and used screen readers to access the computer. For each of the questions in the bottom group, the answers were selected from a 5-point scale. In that group, 6 users expressed at least a moderate desire to make maps; the same 6 rated their mobility skills at least average and also believed that both AudioQuake and LDL's models of navigation were at least moderately relevant to real-world navigation. Of the participants, 3 had extensive experience of other 3D accessible games[22]. Most users had extensive experience of Interactive Fiction which, it was hoped, should make connections and compass directions easier (section 4.2.3 shows that this may be worth investigating further).

[22] though one of these pointed out that in fact there are no other truly 3D accessible games currently available (further clarification was sought and this user had in fact played other accessible games we classify as 3D, such as Shades of Doom).

4.2.2 Routing and Architecture

The users were also asked how they conceptualised both a single route (e.g. to the local shop) and a situation involving multiple routes that might cross (e.g. between buildings on campus). For the single route situation, most (5) said they could imagine it all at once as if they had a map, as opposed to a series of instructions to be carried out at specific points. For the multiple routes situation, only 4 said they could imagine the routes simultaneously and the rest had to specifically remember where the routes cross.

These questions were designed to see how easy it may be for the users to specify a map as a series of related routes, via the connections between rooms (in which case they would need to be fully aware of where the routes through the level cross). There is some contention as to whether prior visual experience is a prerequisite for good spatial awareness; different studies have drawn differing conclusions [8].

The question of whether the current highest (3D layout) layer is still too concrete was put to the users. They were given a choice of how they would like to specify the relationships between rooms, either: using compass points as it is now; simply specifying that rooms are connected and having the LDL system do the layout and a hybrid of the two. Equal numbers of users (3) sided with the two different approaches and one voted for the hybrid. This tells us that implementing a higher-level layout feature would be welcomed, though the challenge—effectively 3D graph layout with constraint satisfaction—is significant.

It should be noted that one thing users *did not* ask for was any more complex room shapes than the cuboid. They also did not request the ability to specify arbitrary angles for connections between rooms. This is likely to be because no current accessible games provide these features except for AudioQuake (by means of already-made maps) and navigation of such structures has already been noted to be difficult by AudioQuake players on the AGRIP mailing list.

4.2.3 Causes of Errors

The users were asked to attribute various potential causes of errors. The results are shown in table 2.

Only one out of the 7 users gave up on using the system. There was unfortunately no reason given. However, this user was the only one that had (a) not played other accessible games or any Interactive Fiction and (b) had rated their mobility ability below average. It would be interesting to carry out a wider study to find out if there is a correlation between spatial awareness and ability to navigate in games (and, if so, can one reinforce the other?); there is already some research that suggests this may be possible [11].

Table 2: Attribution of errors across all users.

Error type	Users
Vision impairment	4
Human error	4
XML unfamiliar	4
XML too strict	2
Conceptualising layout	2
Spatial awareness	3
Missing features	2
Others	1

Table 3: Alternative interfaces.

Interface type	Votes
XML	3
Natural language	2
Dialogue	3
Non-speech audio	2
In-game	5
Haptic	1

4.2.4 Proposed User Interfaces

Table 3 shows the votes cast for each type of proposed user interface that could be layered on top of LDL. No additional alternatives were suggested by the users. It was noted in the survey that buildings are inherently structured things so "natural language" would still be in a somewhat constrained form, albeit less strict than the current XML. It was expected that users would *not* be so keen to use an in-game interface as this would make it even harder for them to get a global overview of the level.

5 Conclusions

This paper has highlighted the need for an accessible level authoring system and proposed a prototype of such a system that, although currently targeted at vision-impaired people, could be adapted for people with a wide range of disabilities in future. We also believe that the system may have a place with non-disabled individuals who may wish to specify their levels in a more abstract way, or who are working as part of a level design team and may only be responsible for the overall structure, or certain fine details of the level.

The system has enabled blind gamers to create

levels for a mainstream game that, although currently relatively simple in nature, have much room for improvement due to the layered design of LDL. Additionally, classic errors such as leaks have been eradicated. The source code for the levels is machine-readable and writable, meaning that tools can be developed to analyse them and even give the computer the ability to create new levels—possibly on-the-fly and as a result of the player's actions and performance in-game.

5.1 Further Work

We have sought feedback from users and found that the approach seems to be promising. Further development, of LDL and the requested interfaces to it needs to take place, followed by a larger user study. This may enable us to find out more about the possible links between spatial awareness, navigation ability and use of AudioQuake and LDL—perhaps the tools being developed could be used in an educational setting to improve spatial awareness in young blind people.

Future work must also include the development of user interfaces tailored for people with other types of disabilities and an investigation as to whether standard 3D formats such as X3D[23] and COLLADA[24] can be used to improve interoperability.

6 Acknowledgements

The authors would like to thank Sabahattin Gucukoglu of the AGRIP project for his advice and support and the Grundy Educational Trust for part-financing this work.

References

[1] Matthew T. Atkinson, Sabahattin Gucukoglu, Colin H. C. Machin, and Adrian E. Lawrence. Making the mainstream accessible: Redefining the game. In *Sandbox '06: Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames*, pages 21–28, New York, NY, USA, July 2006. ACM.

[2] K. Bierre, J. Chetwynd, B. Ellis, D. M. Hinn, S. Ludi, and T. Westin. Game not over: Accessibility issues in video games. In *Proc. of the 3rd International Conference on Universal Access in Human-Computer Interaction*. Lawrence Erlbaum, 2005.

[3] Johnny Friberg and Dan Gärdenfors. Audio games: new perspectives on game audio. In *ACE '04: Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 148–154, New York, NY, USA, 2004. ACM.

[4] Hiromichi Fukutake, Yoshiaki Akazawa, Yoshihiro Okada, and Koichi Niijima. 3d object layout by voice commands based on contact constraints. In *CGIV '05: Proceedings of the International Conference on Computer Graphics, Imaging and Visualization*, pages 403–408, Washington, DC, USA, 2005. IEEE Computer Society.

[5] James Paul Gee. What video games have to teach us about learning and literacy. *Comput. Entertain.*, 1(1):20–20, 2003.

[6] Dimitris Grammenos. Game over: learning by dying. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1443–1452, New York, NY, USA, 2008. ACM.

[7] IGDA. Accessibility in games: Motivations and approaches. Technical report, IGDA, `http://www.igda.org/wiki/Game_Accessibility_SIG/Papers`, 2004.

[8] J. M. Loomis, R. L. Klatzky, R. G. Golledge, J. G. Cincinelli, J. W. Pellegrino, and P. A. Fry. Nonvisual navigation by blind and sighted: assessment of path integration ability. *Journal of experimental psychology*, pages 73–91, 1993.

[9] Niklas Röber and Maic Masuch. Auditory game authoring. In Quasim Mehdi, Norman Gough, and Gavin King, editors, *Proceedings of CGAIDE 2004 Conference*, November 2004.

[10] Kurt Squire. Video games in education. *International Journal of Intelligent Simulations and Gaming*, 2:49–62, 2003.

[11] D Stanton, P Wilson, and N Foreman. Using virtual reality environments to aid spatial awareness in disabled children. In *The First European Conference on Disability, Virtual Reality and Associated Technologies*, 1996.

[12] Shari Trewin, Vicki L. Hanson, Mark R. Laff, and Anna Cavender. Powerup: an accessible virtual world. In *Assets '08: Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility*, pages 177–184, New York, NY, USA, 2008. ACM.

[13] Aaron Weiss. Desktops in 3d. *netWorker*, 11(1):26–33, 2007.

[14] Thomas Westin. Game accessibility case study: Terraformers - a real-time 3d graphic game. In *The Fifth International Conference on Disability, Virtual Reality and Associated Technologies*, 2004.

[23] `http://www.web3d.org/`
[24] `http://www.collada.org/`