CREATING A VIRTUAL GALLERY FOR THE PRESENTATION OF ARTWORKS

Volodymyr Snihur†,Ivan Bratus †, Anna Gunka †, Denis Sharikov††, Myroslava Perysta††, Halyna Kuzmenko††

† Borys Grinchenko Kyiv University, Ukraine

†† Kyiv Municipal Academy of Circus and Performing Arts, Ukraine

Summary
The article analyzes the concept of virtual gallery as a viable alternative to traditional physical galleries and/or museums. Additionally in this article we give a relatively brief description and our insights from an artist point of view of the technologies and image formats that can be used for such a project. Most of the components are judged on several criteria: ease of use and editing for gallery creator, "gallery hall" versatility, interactive presentation of content, file size and lastly — usage. Questions of hosting and server setup are omitted as they are not essential for this article and lie mostly outside the scope of this text or are more universal variables which are better discussed on a case by case basis.

Key words:
Input here the part of 4-5 keywords.

1. Introduction

Let us begin with the statement that galleries and art as it is are needed for humans, be it species as a whole or as individual entities. Throughout our history we not only have made artistic expression an integral part of our cultures (after all it is the first thing we find in villages or towns made by long gone civilizations of "times of yore" and it will be the defining aspect of our time for future archeologists, at least it should be) but we devised new ways of making it seen: from scribbles on mammoth tusks worn as primitive jewellery to collections housed by those who have money and resources to support them and eventually to the museums, whose sole responsibilities are preservation, classification and research. Artists, engineers, musicians, writers, poets, actors… everyone who has ever made something they wanted to be seen by anyone else has always wanted a place to show their works, yet museums only house works of "significant value", be it cultural, monetary, scientific or historical. Private galleries however have less limitations on what they accept, who can use their "platform" and influence (of course the more prestigious gallery is — the pickier it will be) to make their name heard and achievements seen. Traditional approach is that such endeavors (for-profit, non-profit, sole proprietorships or non-commercial personal collections) have to rent, buy or use existing property on which they'll operate, wepropose to change it. Our approach does not rely on physical space, nor is it necessary to have artworks in gallery's possession as they will stay with their authors, therefore easing the responsibility, liability, transportation and operational costs for both parties involved.

Manuscript received October 5, 2021
Manuscript revised October 20, 2021
https://doi.org/10.22937/IJCSNS.2021.21.10.28

2. Theoretical Considerations

When making such a gallery we need to consider two primary factors influencing our choice of style and technology used: devices currently in use by target demographic and rendering technologies available.
While most people now tend to use mobile phones, aka — smartphones, powered by either Android, iOS, Linux or their various derivatives (while market for purely Linux-powered phones is almost non-existent, in future years it may increase by substantial amount purely because Linux itself offers greater flexibility and is not as locked-in as even "stock" Android), there are various other PCs, laptops, and even gaming consoles capable of installing and launching "native" or "web" apps, which again fall into different categories even if we were to divide them only by types of installed operating systems, with Linux derivatives making an impressive list on their own.
While the former category is what may be most common technology in everyday use, the latter has greater 3D rendering capability and does not suffer from user input limitations, so common on touch-screen devices. Thus we may disregard mobiles in this discussion, although one of the later paragraphs of this article will give a more in-depth explanation as to what problems will inevitably arise if we were to make a gallery or any such project with the focus solely on mobile and various hand-held devices.

Available image formats. In the next paragraphs we will briefly go through the technologies that could enable such a project to function as it should. It is also worth pointing out that most of them are not suitable for a fully immersive presentation of arts. There are many alternative formats like GIF[1], BMP[3][4], PNG[5], WEBP[6] or JPG[7] but not all of them would be suitable for a Wordpress website,

much less so for a gallery like this one. Evaluation will be done by several criteria such as ease of editing and displaying, file size, compression artefacts. For file size test a picture of uniform color (white) with dimensions of 400 by 400 pixels will be used to give compression algorithms the easiest to complete task and as a result — most reliable, albeit heavily detached from reality results.

Firstly we will look at the image format used since the early 90th — GIF.[1] This format is currently the only one with full support for animations which would be quite helpful for lightweight 3D overview photoshoots like if we were walking around a statue for example. Is widely supported by devices, OSs and various programs, additionally it is considered one of the formats capable of retaining high detail in static (non-animated) images. Primary use is still mostly limited to animations or video clips that absolutely should be seen and/or looped endlessly. On the other hand we are losing sound data while converting things from .mp4 or .avi to .gif file (provided it is one of the situations where such conversion is deemed a good alternative to usual video file). One more issue with this format is that it can display no more than 256 colors, so everything in the usual range of "16 million colors" gets reduced to just 256 different variants and this is why most multi-color image animations have pretty bad overall quality. Despite this oddly enough, .gif is still sometimes recommended as a way to store high resolution photos, or at least for use in various blogs and social media. The test picture is between 0.6 and 1.5 kilobytes in size (tested with Gimp 2.10.24[2] and Microsoft Paint). Admittedly Gimp is more successful in this regard, whereas Paint managed to triple file size.

Bitmap[3][4] files are often used for even greater detalisation, partly because of its "lossless" image compression (actually it has different compression variants, but default option is "no compression"). The file itself in its simplest form can be described as an array of raw data about pixel color values, which in turn make up an image. Great for creating digital artworks and not so good for displaying them, in no small part because of the huge size of image files. For example, saving the same file from the previous test, results in a file with 462 kilobyte file size (file saved as 24-bit bitmap file, which is a default setting unlikely to be changed in normal everyday use). This was just a plain white 400x400 pixel area which is… tiny in comparison to standard not-so-good photos taken on a 1.5 megapixel camera from several years back, all the while file size is almost half of the really bad color photo taken two years ago.

Another possible alternative is .png (Portable Network Graphics)[5] — in a way a replacement and an evolution of GIF format. For one it does not support traditional printing color space (CMYK), which is not an issue as most printers by now at least somewhat can translate RGB values to CMYK for printing. Secondly it is a lossless file format which would be perfect for an art showcase and is designed to be used for network applications like websites, which means it should be easier to work with from this standpoint. Saving the same test image however results in 1.44 to 1.49 kilobyte size. Which is honestly impressive and much, much better result compared to all the rest file formats we've discussed earlier if we were judging them on the file size alone.

Webp[6] format developed by Google in 2010 with the aim to "create files that are smaller for the same quality, or of higher quality for the same size, than JPEG, PNG, and GIF image formats". This format is surely developed for web applications first and foremost and is indeed better for tightly filled with images portfolio websites, news pages or gaming forums. On the other hand usefulness of this format is limited by its own area of use: it can hardly be opened with anything but browsers, Android/iOS gallery or some image editing tools, which means that once saved image can hardly be changed (artists should always, always have their master-files untouched for this very reason) or displayed in a more traditional way like automatic photo frame or a gallery (not even talking about the fact that image format itself is pretty much irrelevant here, beyond file size and quality). Webp can't be used in game engines or 3D software, it shouldn't be anyway as there are way better variants for that (.dds (Direct Draw Surface) .png .bmp .jpg as either stand-alone textures or as texture/normal images for materials). Saving the same test square in Gimp in lossless mode results in 746 kilobyte file, lossy compression brought size down to just 1.01 kilobyte. Non-3D Paint does not have an option to save as .webp.

Last but not least and definitely favorite among general populace format is .jpg or .jpeg[7] (three-letter file extension is a relic of old days when Microsoft OS had a hard limit of no more than three characters for file extensions, nowadays both variants are used interchangeably) which is mostly lossy format, although it does support lossless compression. It is easier to notice compression in the form of squares of different color appearing upon zoom-in, such artifacts and many others are introduced more heavily the more aggressive the algorithm used or if image is compressed multiple times, which leads to overall color degradation and quality reduction. For thumbnails it is acceptable to have very bad but mostly fine image quality (we do not expect to be able to see great detail in a square roughly up to two centimeters in size) which means heavy cropping and something like 100:1 compression ratios are allowed. However photos, scans and

digital artwork should be saved as close to original resolution as possible as we can always compress, crop and edit images as we see fit, while up-scaling the same image is extremely hard even if new (mostly experimental) AI image editors are used. Test image on save comes up in 3.03 and 3.14 kilobyte variants saved in Gimp and Paint respectively. Curiously enough, increasing quality from "acceptable" 90 to "perfect" 100 in Gimp has no effects on file size. For comparison, a properly compressed 4 color 750x1125 pixel image saved in .jpg format is barely 284 kilobytes in size with negligible quality reductions mostly visible as slightly blurred edges if we zoom in very closely.

Interactive part of the project. Having decided on what image format we would like to use we have to think about what technology, or rather the way of presenting art, can be used in our project. To name a few already available, albeit slightly outdated and/or inconvenient ones we have: Flash[8] animation, Three.js[9] and WebGL.[10] Not accounting for usual animations and portfolio websites, which however now are less common in non-professional circles as platforms like ArtStation, Facebook and Instagram have replaced most of traditional personal web pages, much in the same way as Reddit came in to replace Digg, which in turn was a replacement for many specialized forums.

While we could even use video presentation and it would largely be fine as is, the aim of this project is to create an immersive life-like exhibition hall while using as little computing power as possible and being relatively user-friendly while doing so. Additional criteria include ease of change or setting up and flexibility in creation of different exhibits. Thus any less "technology demanding" ways like above mentioned video presentation sadly do not fit as the main solution, merely a part of it to be used only when and if necessary.

Flash animations despite being an old way of presenting animations, created by Macromedia, which was bought by Adobe in 2005.[8] Flash along with Dreamweaver and HomeSite were merged into Adobe software suite with Flash itself being used in one of their 2D animation products — Adobe Animate, as an export format up until Adobe Flash was to be deprecated in favor of HTML5 or WebGL with Adobe Animate kept as a part of Adobe suite even after this switch. This technology was what started an era of modern web animation and allowed the creation of simple 2D and sometimes even 3D games, which could be played directly in the browser window. Most of said games have been preserved by Kongregate or Internet Archive where and if it was possible. For our use however Flash would hardly be suitable as it is not only deprecated format with much support removed in Windows 10, Google Chrome, Firefox or any other browser and/or Linux distributions (at least it was recommended to fully remove any and all Flash components by end of 2020, this however likely does not account for older systems as, for example devices using Android 5 still have built-in Flash support, unless app responsible for it was removed, deactivated or browsers have since stopped recognising it, which is partially unlikely as it was made system component and thus undeletable without root access into system memory partition) the technology itself was fairly limited, required special app to create and sometimes to be displayed to intended audience or have enabled Flash support in browser or separate browser extension present on device. Additionally Flash did not allow for large and overly interactive content, although there were a lot of games created during this period.

Three.js is a JavaScript library purpose-built for creating fairly lightweight animations, mini-games and physics simulations [9]. Basically it is a mini-framework for anything animated and is even used for promotional materials (because of course it would be). Currently it can be self-hosted on personal or rented server or on Threejs.org own servers, most of the documentation is readily accessible and code is open sourced under MIT License on GitHub and latest stable version is downloadable in zip archive from Threejs.org website as well as GitHub repository. It has its own editor, which is fairly limited if compared with other similar editor components, let alone — game engines, but in turn with a bit of coding and diving into their tutorials we can create complex animations, small games and even physics simulations (example: cloth simulation present as an example on their website).

Alternative and perhaps, the last one suited for our use is WebGL.[10] Here it is listed separately mainly because this format is supported by many more editors, game engines and websites like itch.io — indie game and asset collection/store. Oddly enough the standard is born partly as a successor and a replacement for Flash (at least in eyes of consumers), partly as an evolution of OpenGLES, which is itself a specialized variant of OpenGL designed to be used in embedded systems (mainly smartphones and browsers) and is supported by both AMD and Nvidia. Partially supported by Apple, Mozilla, Google in the form of their browser development departments with Khronos Group seemingly leading development and maintenance efforts. When Three.js is a WebGL-specialized library with a built-in editor capability, it is but one of the many specialized tools one can use for WebGL development, albeit it is one of if not the most optimized of all.

Our primary interest here is the format's ability to be used anywhere and on any device which supports the standard (ideally support would be on hardware level) and has a compatible browser. In fact most modern smartphones and GPUs meet our requirements and even browsers such as Internet Explorer do support WebGL. Next requirement would be ease of editing scenes or "exhibition rooms", 3D space creation and interactive content showcase, all of which are possible.

Downsides however are that tools we can use are fairly limited in number and optimization of content, the level or scene is quite resource-demanding on PCs, especially on low-end ones. Some computers and mobiles can not load the respective scene at all and one of the test machines (which is fairly old and is used solely for testing accessibility or optimization) had to be rebooted twice and once rebooted on its own. In the end that machine spent around 11-17 minutes to load our first test scene and then it crashed. Yet another downside is that graphics, mainly textures, are the heaviest and most demanding part of the whole package, not including underlying physics engine, and have to be compressed till acceptably size/quality ratio is met. Not the least factor here is image format, which was discussed quite briefly in previous part of the article, that said .jpg and .png may be the best pick for this project due to their file size, ease of importing, editing and generally widespread use. Third and final major negative aspect of using this format is that it essentially relies on browser for content rendering, which in combination with fairly limited access to GPU and RAM, need to reload, compile shaders and spatial materials and render the scene on each and every page reload or level transition (this however can be somewhat mitigated by the fact that some assets can be pre-loaded and pre-rendered or shared between different scenes, or as an extreme measure — loaded as single resource file, similar to atlas textures) means that we are limited in what, how much, when and how we want to show not only because the machine our users have is not top-tier gaming PC but the browser itself is an inferior platform for such content.

This approach is currently the best known to us if we were to make such a gallery that exists as a pseudo-3D space, accessible through online means and enables full-resolution (or acceptable quality/file size/load time) interactive view of artworks in 2 or 3D. As an additional benefit, this also enables us to make specialized exhibition halls for each author or experiment with the concept of "exhibition hall" itself. Yet another benefit is that simple copying or downloading of pictures, 3D models, sounds or videos used is made much harder for an average end-user, as such the art is much safer than if it was displayed on Wordpress blog or traditional museum website, should its author be concerned with this specific question.

Technical characteristics. Actor movement and camera control is handled by the script provided below. This is fairly close to implementation of movement in games, so it is only fitting to reference documents from actual game engine.[11]

```
var direction = Vector3()
if Input.is_action_pressed("ui_up") ||
Input.is_key_pressed(KEY_W):
    direction += -global_transform.basis.z
if Input.is_action_pressed("ui_down") ||
Input.is_key_pressed(KEY_S):
    direction += global_transform.basis.z
if Input.is_action_pressed("ui_left") ||
Input.is_key_pressed(KEY_A):
    direction += -global_transform.basis.x
if Input.is_action_pressed("ui_right") ||
Input.is_key_pressed(KEY_D):
    direction += global_transform.basis.x
    direction = direction.normalized()
```

Input buttons are mapped to UI control variables in case these same directional inputs will be used in interactive menus (from usability standpoint it will be much easier for players/visitors to use controls they are already familiar with and have their fingers on, which is part of the reason why many games map frequently used UI actions to left part of keyboard as closely to WASD movement keys as possible) as well as checked through the literal key name to avoid different user input mapping in case UI actions are set to different set of keys, like for example arrow keys.

The actual inputs however are taken only when and if the mouse pointer is captured by program:

```
if Input.get_mouse_mode() ==
Input.MOUSE_MODE_VISIBLE:

Input.set_mouse_mode(Input.MOUSE_MODE_
CAPTURED)
```

MOUSE_MODE_VISIBLE is a check of mouse pointer visibility (to user) which returns true/false value depending on the pointer state. MOUSE_MODE_CAPTURED sets mouse state so that the cursor is captured by the app and mouse is unable to leave the window, which is especially useful when the app's effective area is larger than the area visible to the user. As is the case with our gallery — effective 3D space area is much larger than what can be shown in one frame.

The code responsible for the 3D navigation and viewpoint direction change can be condensed to this:

```
if event is InputEventMouseMotion and Input.get_mouse_mode() == Input.MOUSE_MODE_CAPTURED:
    rotate_y(-event.relative.x * mouse_sensitivity)
    $Pivot.rotate_x(-event.relative.y * mouse_sensitivity)
    $Pivot.rotation.x = clamp($Pivot.rotation.x, -1.2, 1.2)
```

Clamping rotation.x sets limits on how far up and down camera can pivot. If we were to remove limits the camera could have unlimited 360 degrees movement, -1.2, 1.2 clamping value sets realistic life-like limits.[12]

On collision with “Exit” area, gallery releases mouse and closes program environment.

3. Conclusion

We have just briefly discussed possible ways of creating proposed online gallery and provided code bits that should serve most functions of bare-bones proof of concept project. Practical uses however were not discussed and are left for future research articles in part due to the possible range of applications being way bigger than it was originally anticipated at the time of writing this article.

References

[1] W3 GIF Specification:
https://www.w3.org/Graphics/GIF/spec-gif87.txt

[2] GIMP: https://www.gimp.org/

[3] Simplified Bitmap Specification:
https://cdn.hackaday.io/files/274271173436768/Simplified%20Windows%20BMP%20Bitmap%20File%20Format%20Specification.htm

[4] Library of Congress archival Bitmap Specification:
https://www.loc.gov/preservation/digital/formats/fdd/fdd000189.shtml

[5] W3 PNG Specification: https://www.w3.org/TR/2003/REC-PNG-20031110/

[6] Google Webp Container Specification:
https://developers.google.com/speed/webp/docs/riff_container

[7] W3 JPEG Specification:
https://www.w3.org/Graphics/JPEG/jfif3.pdf

[8] Adobe Flash technical archive:
https://helpx.adobe.com/air/archived-docs-download.html

[9] Three.js Documentation:
https://threejs.org/docs/index.html#manual/en/introduction/Creating-a-scene

[10] Khronos Group documentation on WebGL:
https://www.khronos.org/webgl/

[11] Godot Docs guide to in-game scripting:
https://docs.godotengine.org/en/stable/getting_started/scripting/index.html

[12] Godot Docs in-depth explanation of Euler angles and camera movement:
https://docs.godotengine.org/en/stable/tutorials/3d/using_transforms.html