AutoGameUI: Constructing High-Fidelity GameUI via Multimodal Correspondence Matching

Zhongliang Tang*, Qingrong Cheng*, Mengchen Tan*, Yongxiang Zhang, Fei Xia†

TiMi L1 Studio, Tencent, China

{clarezltang, kirolcheng, mengchentan, freizhang, wallacexia}@tencent.com

Abstract

Game UI development is essential to the game industry. However, the traditional workflow requires substantial manual effort to integrate pairwise UI and UX designs into a cohesive game user interface (GameUI). The inconsistency between the aesthetic UI design and the functional UX design typically results in mismatches and inefficiencies. To address the issue, we present an automatic system, **AutoGameUI**, for efficiently and accurately constructing GameUI. The system centers on a two-stage multimodal learning pipeline to obtain the optimal correspondences between UI and UX designs. The first stage learns the comprehensive representations of UI and UX designs from multimodal perspectives. The second stage incorporates grouped cross-attention modules with constrained integer programming to estimate the optimal correspondences through top-down hierarchical matching. The optimal correspondences enable the automatic GameUI construction. We create the GAMEUI dataset, comprising pairwise UI and UX designs from real-world games, to train and validate the proposed method. Besides, an interactive web tool is implemented to ensure high-fidelity effects and facilitate human-in-the-loop construction. Extensive experiments on the GAMEUI and RICO datasets demonstrate the effectiveness of our system in maintaining consistency between the constructed GameUI and the original designs. When deployed in the workflow of several mobile games, AutoGameUI achieves a 3× improvement in time efficiency, conveying significant practical value for game UI development.

Introduction

With the rise of personal computers and smart mobile devices, the game industry has generated considerable economic and cultural impacts. However, as a critical step in game development, game UI development still faces many challenges. Although many studies (Li et al. 2022; Warner, Kim, and Hartmann 2023; Jiang et al. 2024) explore UI automation for web pages or mobile apps, few discover and address the issues in game UI development.

In the traditional workflow of game UI development, two specialized teams work concurrently to construct a cohesive game user interface (GameUI). The first team, made up of UI designers, focuses on crafting visually appealing elements to achieve diverse game styles. The second team, composed of UX designers, emphasizes usability and functionality by prototyping effective logical structures. As illustrated in Fig .1, the distinct focus areas often lead to conflicting design elements and inconsistent layouts, making it difficult to integrate the aesthetic UI design with the functional UX design. Game developers must employ substantial effort to align these differences, such as manually finding correspondence between the UI and UX designs, painstakingly adjusting the position, size, anchor, and asset assignment for the UX elements one by one. This workflow is time-consuming and low-quality, easily causes delays.

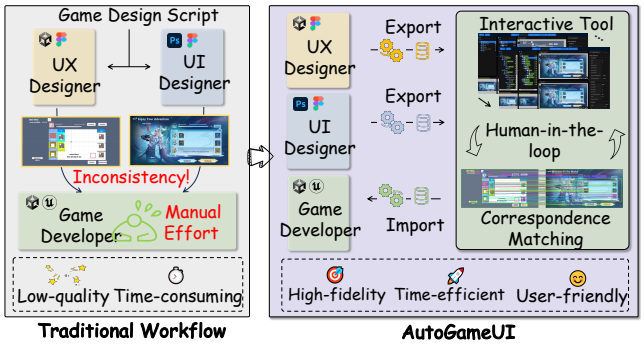


Figure 1: Comparison between the traditional manual workflow and AutoGameUI in game UI development.

To alleviate the bottleneck posed above, this paper introduces AutoGameUI, an automatic system for GameUI construction. As depicted in Fig. 1 and Fig .2, our system comprises two main components: multimodal correspondence matching and an interactive tool. The multimodal correspondence matching is a two-stage learning pipeline, with the first stage focusing on representing UI and UX designs in a latent space, and the second stage aiming to estimate the correspondences between them. Unlike previous studies (Yamaguchi 2021; Bai et al. 2023; Jiang et al. 2024) that examined the representation of graphical UIs from a limited number of perspectives, our method considers various aspects, such as spatial layout, functional semantics, textual content, hierarchy, and even rendering order. Moreover, some works (Wu et al. 2023; Otani et al. 2024) attempted to directly esti-

*These authors contributed equally.

†Corresponding author.

mate correspondences between pairs of graphical UIs based on one-to-one element similarity. In contrast, we adopt the divide-and-conquer strategy to formulate a hierarchical correspondence matching problem. We utilize grouped cross-attention (GCA) modules to compute one-to-group similarity, which, further combined with a constrained integer programming algorithm, enables us to estimate optimal correspondences efficiently. Following prior works (Baciková, Porubän, and Lakatos 2013; Panayiotou et al. 2024), we design a universal protocol to describe the UI and UX designs, enabling the data stream across different platforms. Using the universal data protocol and the optimal correspondences, we can achieve high-fidelity effects in the GameUI construction. To enhance user experience, we develop an interactive tool that provides intuitive feedback and guidance during the construction process. This tool also supports data annotation, allowing us to build the GAMEUI dataset from real-world games for training and evaluation. In summary, this paper makes the following contributions:

- We introduce an automatic system for high-fidelity GameUI construction. This system overcomes the bottleneck of traditional manual workflow and significantly accelerates the game UI development.
- We propose a two-stage multimodal learning pipeline that estimates optimal UI/UX correspondences from comprehensive representations, incorporating grouped cross-attention modules and constrained integer programming to enhance efficiency and accuracy.
- We implement an interactive tool with a universal data protocol that supports cross-platform applications and facilitates human-in-the-loop construction.
- We present the GAMEUI dataset, the first-of-its-kind dataset that contains paired UI and UX designs with rich-structured data and high-quality annotations.

Related Work

Representation of UIs

Representing graphical UIs is crucial for tasks such as UI recognition (Li et al. 2022), UI completion (Jiang et al. 2024), UI generation (Inoue et al. 2023), and UI structuring (Wu et al. 2021). The essential attributes of graphical UIs include spatial layout, semantics, image texture, textual content, view hierarchy, and rendering order. Most methods primarily focus on spatial layout and semantics but overlook others. LayoutTrans (Gupta et al. 2021), LayoutDM (Inoue et al. 2023) and LayoutFlow (Guerreiro et al. 2024) generated UIs based on layout geometry and semantics. Other efforts (He et al. 2021; Yamaguchi 2021; Jiang et al. 2024) incorporated image texture and textual content but fell short in understanding hierarchy and rendering order. Some studies (Li et al. 2021b; Bai et al. 2023) modeled hierarchy while rendering order remains unexplored.

Correspondence between UIs

Identifying corresponding elements between pairs of UIs has been investigated for some time now. Earlier studies (Kumar et al. 2011b,a) used probabilistic optimization to establish correspondences for automatic design transfer. Dayama et al. (2021) presented an integer programming algorithm to optimize correspondences and developed an interactive tool for layout transfer. LayoutBlend (Xu et al. 2022) found the optimal correspondences between UIs with maintained hierarchical consistencies. Otani et al. (2024) introduced a criterion to measure the similarity between paired layouts. These works are based on heuristic rules, and have poor generalization in practical use.

Some works further applied deep learning methods for correspondence matching. LayoutGMN (Patil et al. 2021) introduced a graph neural network to determine the similarity between arbitrary layouts. Wu et al. (2023) used a transformer model to capture latent representations of layout elements and then infer correspondences between UIs. Most of these methods compute a full similarity matrix between pairwise elements. When the number of elements is large, solving the correspondence problem under additional constraints can become extremely time-consuming.

Artificial Intelligence in Game Industry

AI technology has been widely used in the game industry, involving game-playing bots and procedural content generation. Prior works (Vinyals et al. 2019; Ye et al. 2020; Fan et al. 2022) utilized reinforced AI to develop agents, enhancing collaborative play and NPC control. Generative AI has also been employed to create game assets, such as 2D graphics (Batifol et al. 2025), 3D models (Zhao et al. 2025), music and voices (Takada et al. 2023; Anastassiou et al. 2024), and animations (Liu et al. 2023; Chen et al. 2024).

Some efforts (Li et al. 2021a; Gonçalves et al. 2023) appeared to intersect with human-computer interaction and game development, but their focus has largely been on gameplay and player experience, with limited insights into game UI development. Moreover, intelligent UI development remains in its early stages. Existing tools such as Uizard (2024) and Framer (2025) are primarily designed for web or mobile apps, and perform poorly in real-world game UIs, which often feature aesthetics and complex layouts.

GameUI Construction

This section outlines our system illustrated in Fig. 1. We begin by factorizing UI and UX designs into multiple modalities with discrete parameters. Then, we use self-supervised learning to generate continuous multimodal representations and formulate a top-down hierarchical correspondence matching problem. To address it, we introduce the grouped cross-attention module to learn matching probability and apply integer programming for optimal correspondences. With the correspondences, we design a protocol to integrate UI and UX designs. Lastly, we develop a web-based tool to help users interactively build GameUI.

Learning Multimodal Representation

Data Attributes UI and UX designs are typically organized as trees with attributes such as spatial layout, image texture, textual content, hierarchy, rendering order, and semantics. Spatial layout describes the 2D geometry of both

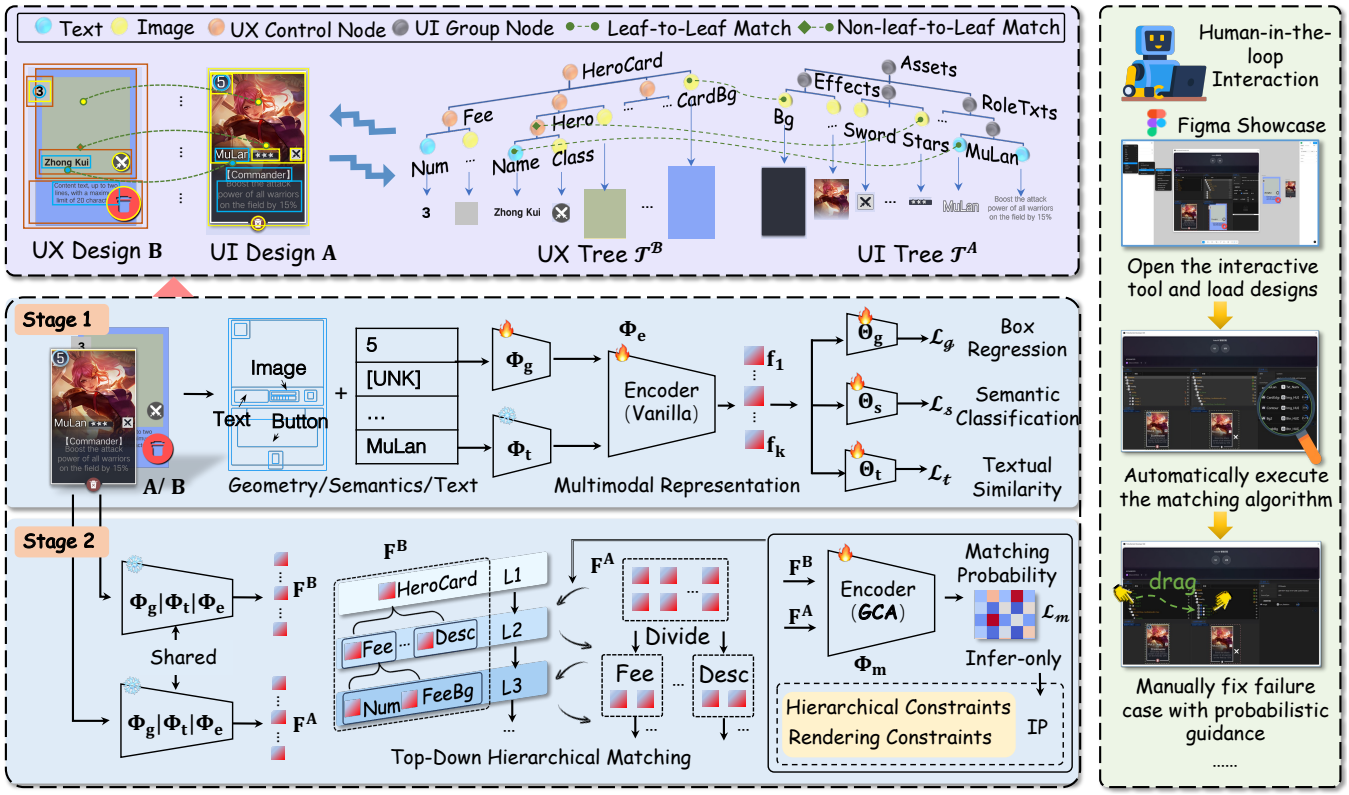


Figure 2: Overview of the AutoGameUI system. Left: multimodal correspondence matching between pairwise UI and UX designs. The first stage focuses on acquiring the multimodal representation of matchable nodes in UI and UX designs. The second stage aims to estimate the optimal correspondences between matchable nodes. Right: interactive showcase in Figma. The user launches the web tool, initiates automatic matching, and manually refines results with matching probabilities.

leaf and non-leaf nodes, while image texture and textual content are renderable attributes found only in leaf nodes. Unlike previous works (Li et al. 2022; Wu et al. 2023; Jiang et al. 2024), we exclude image texture, as color differences between UI and UX designs may introduce noise. Hierarchy shows the organization of nodes, with higher-level nodes covering broader scopes. Rendering order determines the display priority of nodes, and semantics indicate node types such as TEXT, IMAGE, or BUTTON.

In this paper, we represent the tree as $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Each node v_k is defined as (g_k, s_k, t_k), denoting its 2D geometry, semantic label, and textual content. Each directed edge e_{ij} from v_i to v_j is defined as $(\Delta g_{ij}, \Delta h_{ij}, \Delta r_{ij})$, capturing the relative geometry, hierarchy, and rendering order between nodes. We adopt the identical formula in LayoutGMN (Patil et al. 2021) for g_k, s_k, and Δg_{ij}. Specifically, $\Delta h_{ij} \in \{0, 1, 2\}$ indicates whether v_i is an ancestor (0), descendant (1), or unrelated (2) to v_j in the hierarchy. $\Delta r_{ij} \in \{0, 1, 2\}$ indicates if v_i overlaps and is rendered after v_j (0), before v_j (1), or does not overlap (2).

With the definition, UI and UX designs can be denoted as $\mathcal{T}^A$ and $\mathcal{T}^B$. The key difference between $\mathcal{T}^A$ and $\mathcal{T}^B$ is that only leaf nodes are matchable in $\mathcal{T}^A$, while both leaf and non-leaf nodes are matchable in $\mathcal{T}^B$ (see Fig. 2, top). This is because non-leaf nodes in UI are just groups of renderable resources without visual attributes, whereas in UX, non-leaf nodes are controllable widgets or components that can be integrated into GameUI.

Multimodal Representation We utilize a graph neural network $\mathbf{\Phi_g}$ (Patil et al. 2021) to encode the parameters s_k, g_k, and Δg_{ij}. In addition, we use a pre-trained language model $\mathbf{\Phi_t}$ (Reimers and Gurevych 2019) to encode the parameter t_k and adopt the special token [UNK] to pad nodes without text. These heterogeneous features are projected into a unified feature vector, which is further processed by a transformer encoder $\mathbf{\Phi_e}$. With rotary positional encoding (Su et al. 2024), $\mathbf{\Phi_e}$ yeilds a 512-dimensional multimodal representation $\mathbf{f}_k$. Notably, the edge parameters Δh_{ij} and Δr_{ij} are not embedded at this stage and will be discussed in the following section.

We train $\mathbf{\Phi_g}$ and $\mathbf{\Phi_e}$ in a self-supervised manner with three MLP heads: semantic classification $\mathbf{\Theta_s}$, bounding box regression $\mathbf{\Theta_g}$, and textual similarity $\mathbf{\Theta_t}$. The total loss is:

$$\mathcal{L} = \lambda_s \mathcal{L}_s + \lambda_g \mathcal{L}_g + \lambda_t \mathcal{L}_t. \quad (1)$$

$\mathcal{L}_s$ is the cross-entropy loss between $\mathbf{\Theta_s}(\mathbf{f}_k)$ and the semantic label s_k. $\mathcal{L}_g$ combines ℓ_1 and generalized IoU losses for $\mathbf{\Theta_g}(\mathbf{f}_k)$ and g_k. and $\mathcal{L}_t$ is a contrastive loss computed with $\mathbf{\Theta_t}(\mathbf{f}_k)$, $\mathbf{\Phi_t}(t_k^+)$, and $\mathbf{\Phi_t}(t_k^-)$. t_k^+ and t_k^- are positive and

negative text samples. Hyper-parameters λ_s, λ_g, and λ_t balance the importance of the modalities, set to 0.5, 1.0, and 0.1 in experiments.

Learning Correspondence Matching

Objective and Constraints We formulate the correspondence matching problem as follows:

$$\begin{aligned} &\underset{\mathbf{P}\in\mathbb{R}^{m\times n}}{\operatorname{argmin}} \sum \mathbf{P}\odot\mathbf{C}+\mathbf{\Omega}(\mathbf{P},\mathbf{C}), \\ &\text{subject to } \mathbf{P}\cdot\mathbf{1}_n=\mathbf{r},\ \ \mathbf{P}^{\mathrm{T}}\cdot\mathbf{1}_m=\mathbf{c}. \end{aligned} \tag{2}$$

m and n are the number of matchable nodes in $\mathcal{T}^A$ and $\mathcal{T}^B$. $\mathbf{P}\in\mathbb{R}^{m\times n}$ is a binary assignment matrix indicating node matches. $\mathbf{C}\in\mathbb{R}^{m\times n}$ is the cost matrix, with lower values for higher similarity. The feasible space of $\mathbf{P}$ is defined by integer vectors $\mathbf{r}$ and $\mathbf{c}$ (lengths m and n, values between 0 and 1). $\odot$ denotes the element-wise product, and $\mathbf{\Omega}$ is the regularization function to reduce suboptimal matches.

To minimize Eq. (2) efficiently, we exploit a novel transformer encoder $\mathbf{\Phi_m}$ and an integer programming solver. $\mathbf{\Phi_m}$ is equipped with GCA modules to determine unknown matrices $\mathbf{C}$, $\mathbf{r}$, and $\mathbf{c}$. The integer programming solver optimizes the assignment matrix $\mathbf{P}$ constrained by $\mathbf{\Omega}$.

Grouped Cross-Attention Module Unlike previous works (Xu et al. 2022; Wu et al. 2023), we avoid computing the node-to-node similarity matrix directly. This is due to the quadratic growth of penalty terms in $\mathbf{\Omega}$, which makes Eq. (2) computationally infeasible for large numbers of nodes. Instead, we adopt the divide-and-conquer strategy, performing hierarchical matching from top to bottom. We review the hierarchy of $\mathcal{T}^B$ and partition all nodes into several groups based on their association with next-level nodes. Each next-level node and its descendants form a group. This node-to-group matching allows us to recursively estimate correspondences between $\mathcal{T}^A$ and $\mathcal{T}^B$, significantly reducing computational complexity.

Taking the first hierarchical matching as an example, trees $\mathcal{T}^A$ and $\mathcal{T}^B$ are input into frozen first-stage models to obtain 512-dimensional representations: $\mathbf{F}^A\in\mathbb{R}^{m\times 512}$ and $\mathbf{F}^B\in\mathbb{R}^{n\times 512}$. As shown in Fig. 3, we assume that $\mathcal{T}^B$ has L secondary-level nodes, $\mathbf{F}^B$ can be partitioned into L groups, one of which is denoted as $\mathbf{G}_l^B$ $(1\le l\le L)$. Each group $\mathbf{G}_l^B$ and the full $\mathbf{F}^A$ are iteratively processed by a shared cross-attention module, producing $\mathbf{O}_{\mathbf{G}_l^B}$ and $\mathbf{O}_{\mathbf{F}_l^A}$ at the l^{th} iteration. After all iterations, we obtain grouped outputs: $\mathbf{O}_{\mathbf{G}^B}\in\mathbb{R}^{n\times 512}$ and $\mathbf{O}_{\mathbf{F}^A}\in\mathbb{R}^{L\times m\times 512}$. Using index mapping, $\mathbf{O}_{\mathbf{G}^B}$ is added to $\mathbf{F}^B$ for updating:

$$\mathbf{F}^B_{updated}=\mathbf{F}^B+\mathbf{O}_{\mathbf{G}^B}. \tag{3}$$

In contrast, $\mathbf{O}_{\mathbf{F}^A}$ requires additional computation to update $\mathbf{F}^A$. We exploit a two-layers MLP head $\mathbf{\Theta_m}$ to compute a lower-dimensional matrix $\mathbf{M}\in\mathbb{R}^{L\times m}$ as:

$$\mathbf{M}=\mathbf{Sigmoid}(\mathbf{\Theta_m}(\mathbf{O}_{\mathbf{F}^A})), \tag{4}$$

where $\mathbf{M}$ can implicitly represent the matching probability between $\mathbf{F}^A$ and each $\mathbf{G}_l^B$. This is due to cross-attention, where more relevant $\mathbf{G}_l^B$ passes more messages into $\mathbf{O}_{\mathbf{F}_l^A}$,

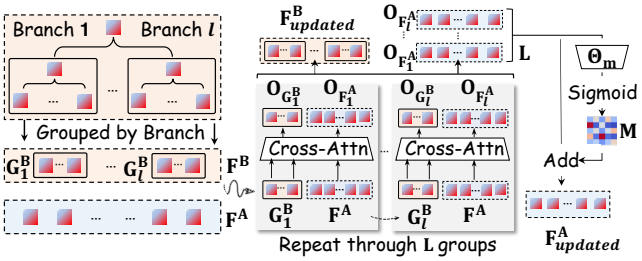


Figure 3: Illustration of the grouped cross-attention module.

resulting in distinct significance in $\mathbf{O}_{\mathbf{F}^A}$ along the first dimension L. Consequently, the cost matrix $\mathbf{C}$ in Eq. (2) is obtained as $\mathbf{C}=\mathbf{1}-\mathbf{M}^\top$, and $\mathbf{F}^A$ is updated by weighted addition as:

$$\mathbf{F}^A_{updated}=\mathbf{F}^A+\sum_{l=1}^{L}\hat{\mathbf{M}}_l^\top\odot\mathbf{O}_{\mathbf{F}_l^A}, \tag{5}$$

where $\hat{\mathbf{M}}=\frac{\mathbf{M}_l}{\sum_{l=1}^{L}\mathbf{M}_l}$ is the proportional normalization of $\mathbf{M}$, with higher probabilities yielding higher weights.

Finally, we incorporate GCA modules into the transformer encoder $\mathbf{\Phi_m}$ and train it with a binary cross-entropy loss $\mathcal{L}_m$. To handle unmatchable cases, we set unmatchable labels as zero-filled vectors during training and filter them out with a threshold $\sigma=0.5$ during testing. This threshold also helps determine the values of matrix $\mathbf{r}$ and $\mathbf{c}$ in Eq. (2) by comparing the matching probability with σ.

Constrained Integer Programming Although the matrices $\mathbf{C}$, $\mathbf{r}$, and $\mathbf{c}$ have been specified, the regularization function $\mathbf{\Omega}$ is not determined yet. The purpose of $\mathbf{\Omega}$ is to penalize suboptimal matches that violate either hierarchical or rendering constraints. Specifically, suboptimal matches occur in cases such as: (1) a node v_i that was originally an ancestor of v_j becomes a descendant after matching (i.e., Δh_{ij} changes from 0 to 1), or (2) the rendering order between overlapping nodes is reversed (i.e., Δr_{ij} changes from 0 to 1). We can define $\mathbf{\Omega}$ as follows:

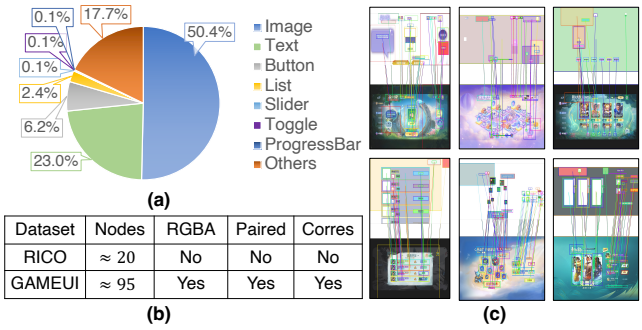
$$\mathbf{\Omega}(\mathbf{P},\mathbf{C})=\sum_{\mathbf{P}_{ii'}=1,\mathbf{P}_{jj'}=1}\tau\cdot(\mathbf{C}_{ii'}+\mathbf{C}_{jj'}), \tag{6}$$

where $\mathbf{P}_{ii'}=1$ and $\mathbf{P}_{jj'}=1$ together indicate the suboptimal matches, and τ controls the penalty strength. Utilizing the edge parameters Δh_{ij} and Δr_{ij}, $\mathbf{\Omega}$ helps maintain consistency in hierarchy and rendering order.

Interactive Construction

Universal Data Protocol Based on the estimated correspondences, we construct GameUI by integrating the visual attributes of UI nodes into the UX nodes. This involves resizing nodes, updating text content, font, and color, and assigning image assets. To facilitate this integration, we introduce a universal protocol.

The protocol has two main components: language and compiler. The language defines node entities from low- to high-level semantics, each with attributes such as position,



Dataset	Nodes	RGBA	Paired	Corres
RICO	≈ 20	No	No	No
GAMEUI	≈ 95	Yes	Yes	Yes

Figure 4: Illustration of the GAMEUI dataset. (a) Distribution of semantic label. (b) Comparison with the RICO dataset. (c) Examples of correspondence annotation.

rotation, scale, anchor, opacity, image texture, and text font. The compiler parses this language into various programming languages, enabling cross-platform execution. We use Antlr4 (Parr 2013), a powerful parser generator that supports multiple languages, including C++, C#, Python, and TypeScript. With Antlr4, our protocol can be adapted for game engines like Unity and Unreal Engine, as well as DCC software such as Photoshop and Figma.

Interactive Web-based Tool As shown in Fig. 1, we implement a web-based interactive tool that takes UI and UX protocols as input and outputs an updated UX protocol with integrated visual attributes. The tool offers several key features: **Fully-automatic construction** efficiently establishes correspondences and enables pixel-perfect integration from UI design to UX design. **Semi-automatic revision** assists users in resolving mismatches by highlighting inconsistencies and suggesting potential solutions, primarily through the matrix $\mathbf{M}$ in Eq. (4). This enables a human-in-the-loop workflow, allowing users to efficiently identify and address issues based on probabilistic guidance. **Previewing** offers multiple rendering modes to help users understand and validate designs. **Editing** allows users to fine-tune the UX design by deleting or creating nodes, repositioning, and updating semantic types. **Correspondence annotation** helps create a high-quality dataset from real-world games, with cloud backup options. **State Management** supports session caching and reloading, allowing users to refine GameUI incrementally without losing intermediate progress. These features provide a comprehensive solution for GameUI construction, combining human supervision with machine feedback, ultimately enhancing the overall user experience.

Experiments

Datasets

To our knowledge, no existing work directly tackles the GameUI construction problem. Most relevant research, such as UI generation or UI retrieval, primarily focuses on the RICO dataset (Deka et al. 2017) or ENRICO dataset (Leiva, Hota, and Oulasvirta 2020). These public datasets are largely crawled from mobile apps in fields such as communication, medicine, and finance. These datasets are quite simple in aesthetics, lack complex layouts and correspondence annotations. Moreover, their image resources are limited to screenshots instead of renderable RGBA images, making them insufficient for GameUI construction.

In this paper, we present the GAMEUI dataset (see Fig. 4), the first dataset designed to build cohesive game UIs from disparate designs. The dataset is collected from different game projects through extensive manual effort by five annotators over 12 months. It comprises 1660 sets of rich, structured game UIs, with each set including a pair of UI and UX designs, along with hierarchy, layout geometry, semantics, and renderable RGBA images. For evaluation, the GAMEUI dataset is split into 1162/166/332 for training, validation, and testing. To avoid overfitting, we adopt comprehensive data augmentation, including random scaling, translation, deletion, and replacement. We also conduct experiments on the RICO dataset, which contains 35851/2109/4218 samples for training, validation, and testing.

Implementation Details

For experiments on the GAMEUI dataset, we adopt a two-stage training strategy (see Fig. 2). In the first stage, models are trained for 300 epochs with a batch size of 4 and an initial learning rate of 1e-5, reduced by a factor of 0.2 every 50 epochs. In the second stage, we freeze the first-stage weights and continue training for 200 epochs under the same settings. For the RICO dataset, which lacks correspondence annotations, we only train the first-stage model to learn multimodal representations, with a batch size of 64, an initial learning rate of 1e-4, and 50 epochs to ensure fair comparison with other baselines. We use Google OR-Tools (Perron and Furnon 2024) to solve the objective in Eq. (2) and set $\tau = 1.0$. All experiments are run on an NVIDIA GeForce RTX 3080 GPU with an AMD Ryzen 12-core CPU.

Evaluation

Since there are no directly comparable benchmarks, our comparison was against the most closely related works as: **LayoutBlend** (Xu et al. 2022) employs a heuristic approach, relying on spatial layout, semantics, and hierarchy to determine node-to-node similarity, and applies the Hungarian algorithm for node correspondence. **LayoutGMN** (Patil et al. 2021), **LayoutTrans** (Gupta et al. 2021), and **LayoutDM** (Inoue et al. 2023), and **LayoutFlow** (Guerreiro et al. 2024) are DNN-based methods that combine spatial layout and semantics for multimodal representation. These DNN models are individually trained on both the RICO and GAMEUI datasets using contrastive, autoregressive, and generative objectives. We use the trained models to extract latent representations of UI and UX nodes, compute node-to-node similarity, and identify the most similar UX node for each UI node as its correspondence.

We evaluate all methods on the GAMEUI dataset, prioritizing three aspects: correspondence matching accuracy, visual consistency of the constructed images, and time efficiency. For matching accuracy, we report precision (**P**), recall (**R**), and F1-score (**F1**) with a weighted macro-average setting. For visual consistency, we integrate the raw RGBA images with correspondences for rendering, and compare

Method	Train Set	HMA	REG	P↑	R↑	F1↑	RMSE↓	PER↓	TIME (s)↓
LayoutBlend	✗	✗	✗	62.3	68.7	64.5	4.57	4.71	3.87
LayoutGMN	RICO	✗	✗	77.1	74.4	75.3	1.65	1.99	0.26
	GAMEUI	✗	✗	81.6	81.6	81.4	1.22	0.82	
LayoutTrans	RICO	✗	✗	65.4	64.4	64.4	5.55	4.91	0.41
	GAMEUI	✗	✗	71.3	68.0	68.8	2.68	1.38	
LayoutDM	RICO	✗	✗	67.5	65.1	65.7	3.48	3.88	2.95
	GAMEUI	✗	✗	72.0	72.3	71.7	1.89	1.46	
LayoutFlow	RICO	✗	✗	81.0	80.4	80.4	2.19	1.30	2.48
	GAMEUI	✗	✗	83.2	84.0	83.5	1.25	0.91	
Ours	RICO	✗	✗	82.9	83.8	83.2	1.96	1.68	0.76
	GAMEUI	✗	✗	84.6	85.8	85.0	1.23	0.96	
	GAMEUI	✓	✗	87.7	88.3	87.6	0.81	0.44	1.01
	GAMEUI	✗	✓	⊘	⊘	⊘	⊘	⊘	⊘
	GAMEUI	✓	✓	**88.2**	**88.8**	**88.1**	**0.21**	**0.19**	1.83

Table 1: Quantitative results on the GAMEUI dataset. **HMA** and **REG** denote hierarchical matching and regularization. ⊘ indicates a timeout (>10s) caused by applying regularization without hierarchical matching. Note that the second-stage model $\mathbf{\Phi_m}$ with GCA modules is exclusively used when hierarchical matching is enabled.

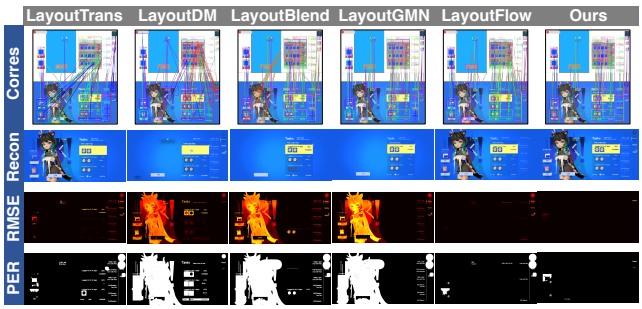


Figure 5: Qualitative results on the GAMEUI dataset.

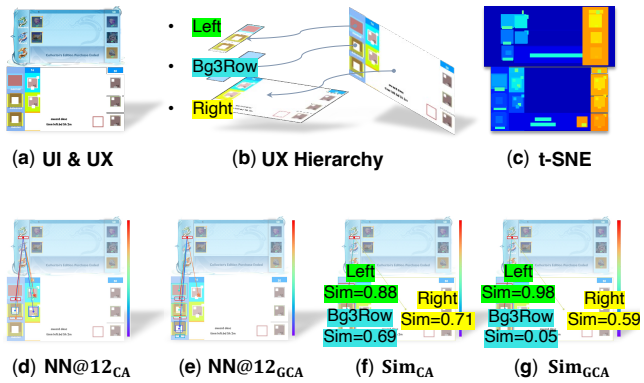


Figure 6: Visualization of the effectiveness of the GCA module. (a, b) show the pairwise UI and UX designs. (c) visualizes the noisy latent representations of UI and UX nodes using t-SNE. (d, e) show the top 12 nearest neighbors of a UI text node in latent space, respectively identified by vanilla cross-attention and GCA. (f, g) show the node-to-group similarity between the UI node and secondary-level UX nodes, with $\mathbf{Sim_{CA}}$ averaged from the node-to-node similarity in vanilla cross-attention.

the rendered image to the original UI design (ground truth). We adopt Root Mean Squared Error (**RMSE**) and Pixel-wise Error Ratio (**PER**) as visual metrics. RMSE measures the average pixel intensity difference (0–255) between the rendered and ground truth images, while PER reports the percentage of pixels with errors. For time efficiency, we record the average time consumption (in seconds) for creating a GameUI from pairwise UI and UX designs.

Results

For matching accuracy and visual consistency, Table 1 illustrates that our first and second-stage models outperform other baselines, generalizing well across RICO and GAMEUI. We also compared GAMEUI against the much larger RICO dataset, where the model trained on high-quality GAMEUI performs better on complex game UIs. Fig. 5 qualitatively reflects our superior performance through the correspondences, the constructed image, and the RMSE and PER maps. Regarding time efficiency, our methods remain competitive. The last column of Table 1 shows that hierarchical matching greatly reduces the computational cost introduced by regularization. Without it, most tasks would exceed the time budget and become impractical for real-world use. Although regularization increases runtime, this trade-off is justified by the significant improvements in matching accuracy and visual consistency.

Furthermore, we analyze the shortcomings of baselines, as shown in Table 1 and Fig. 5. LayoutTrans and LayoutDM perform poorly due to the limited generalization of learnable absolute positional encoding (Vaswani 2017) on complex UIs. In contrast, LayoutFlow's fixed absolute positional encoding (Gehring et al. 2017), though less flexible, is more stable. LayoutBlend fails on complex UIs due to its naive heuristic rules, while LayoutGMN underperforms because it uses fewer modalities and a simpler training strategy.

Here, we study the GCA module and the regularization function. Fig. 6 reveals that positional and semantic misalignment introduces noise, causing vanilla cross-attention to identify incorrect neighbors. The GCA module addresses

Figure 7: Effectiveness of the regularization function. For each τ, the left shows the construction result and the right shows the PER map measured against the UI design.

this by grouping representations using the UX hierarchy and applying attention within each group, which acts as a filter weighted by hierarchical averaging upon vanilla cross-attention. As a result, the GCA module achieves a better similarity distribution and fewer incorrect neighbors. As illustrated in Fig. 7, we validate the effectiveness of the regularization function, as fewer artifacts in visual consistency as the penalty strength τ increases. Overall, these strategies progressively improve correspondence matching and visual consistency compared to the base model.

Ablation Study We conduct an ablation study by independently removing input modalities and regularization constraints. Additionally, we replace the GCA module with vanilla cross-attention instead of removing it entirely, to avoid introducing an inductive bias. We can summarize the experimental results in Table 2 from three aspects:

Input Modalities. *Spatial geometry is the most essential modality for our task.* Removing geometry leads to a dramatic drop in F1 score (88.1 to 42.0) and substantial increases in both RMSE (0.21 to 7.03) and PER (0.19 to 5.56).

Cross-Attention Modules. *The GCA module outperforms vanilla cross-attention.* Replacing the GCA module with vanilla cross-attention results in a decrease in F1 (to 86.0) and increases in RMSE (to 1.22) and PER (to 0.82).

Regularization Constraints. *Rendering constraints have a greater impact on performance than hierarchical constraints.* Removing the rendering constraints leads to a larger decrease in F1 (to 86.6) and greater increases in RMSE (to 1.46) and PER (to 0.63), compared to removing the hierarchical constraints (F1 to 87.8, RMSE to 0.43, PER to 0.23). This suggests that rendering constraints play a more important role, especially in maintaining visual consistency.

Expert Evaluation To evaluate our system with realistic user feedback, we designed an expert-driven study comparing it with the traditional manual workflow. Following common human-centric practices in UI automation, we defined four metrics: Time Efficiency (**TEF**, normalized time statistics, e.g. $\mathbf{TEF}_{trad} = Time_{ours}/(Time_{ours}+Time_{trad})\times 10$), Cognitive Load (**CLD**, perceived mental effort), Output Quality (**OQY**, perceptible visual fidelity), and Subjective Experience (**SJE**, overall satisfaction and ease of use). Except for TEF, all metrics were scored on a 0-10 scale, with

Ablation	F1↑	RMSE↓	PER↓
Full Model	88.1	0.21	0.19
− *geometry*	42.0	7.03	5.56
− *semantics*	87.2	1.05	0.59
− *textual content*	87.5	0.94	0.48
− *rendering constraints*	86.6	1.46	0.63
− *hierarchical constraints*	87.8	0.43	0.23
† *vanilla cross-attention*	86.0	1.22	0.82

Table 2: Ablation study on the proposed model. − represents removing the setting from the full model, and † represents utilizing the setting to replace the GCA module.

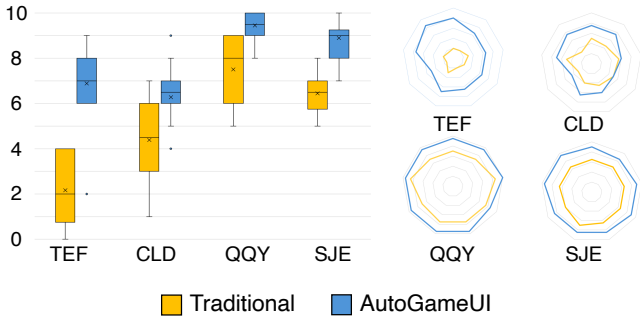


Figure 8: The statistical data of the expert evaluation by comparing our system with the traditional manual workflow.

five levels: Unsatisfactory (0-2), Needs Improvement (2-4), Satisfactory (4-6), Good (6-8), and Excellent (8-10).

Nine experts (5 men, 4 women) were invited to participate in the study. All were familiar with the traditional workflow and had at least five years of professional experience. Each expert was assigned two distinct UI/UX pairs, randomly shuffled to mitigate sequence bias. They used identical desktop PCs and received the same training in scoring criteria and system usage. They could choose which method to try first, but had to complete each task before switching.

The results of this study are shown in Fig. 8. The box plot on the left shows that our system outperforms the traditional workflow across all four metrics, with higher medians and less variability. Specifically, time efficiency improves by up to 3×, and outputs are more visually accurate with few artifacts. The radar charts on the right provide a view to per-user preference, showing that our system achieves consistently high scores among all users, while the traditional workflow exhibits generally lower ratings.

Conclusion

We present a novel system for constructing cohesive game user interfaces, supporting both automatic and interactive pipelines. The automatic pipeline employs a two-stage multimodal learning method to match UI and UX designs, while the interactive pipeline provides multiple assistant features for a user-friendly experience. Extensive experiments on our newly built GAMEUI dataset and the RICO dataset demonstrate the effectiveness. Our system has been deployed in several mobile games, delivering significant improvements.

Acknowledgments

We gratefully acknowledge the continuous support provided by our colleagues Jun Deng and Fei Ling. Their contributions in terms of project guidance and funding have been invaluable to the completion of this work.

References

Anastassiou, P.; Chen, J.; Chen, J.; Chen, Y.; Chen, Z.; Chen, Z.; Cong, J.; Deng, L.; Ding, C.; Gao, L.; et al. 2024. Seed-TTS: A Family of High-Quality Versatile Speech Generation Models. *arXiv preprint arXiv:2406.02430*.

Baciková, M.; Porubän, J.; and Lakatos, D. 2013. Defining domain language of graphical user interfaces. In *2nd Symposium on Languages, Applications and Technologies (2013)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik.

Bai, Y.; Manandhar, D.; Wang, Z.; Collomosse, J.; and Fu, Y. 2023. Layout representation learning with spatial and structural hierarchies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 206–214.

Batifol, S.; Blattmann, A.; Boesel, F.; Consul, S.; Diagne, C.; Dockhorn, T.; English, J.; English, Z.; Esser, P.; Kulal, S.; et al. 2025. FLUX. 1 Kontext: Flow Matching for In-Context Image Generation and Editing in Latent Space. *arXiv e-prints*, arXiv–2506.

Chen, R.; Shi, M.; Huang, S.; Tan, P.; Komura, T.; and Chen, X. 2024. Taming Diffusion Probabilistic Models for Character Control. In *ACM SIGGRAPH 2024 Conference Papers*, 1–10.

Dayama, N. R.; Santala, S.; Brückner, L.; Todi, K.; Du, J.; and Oulasvirta, A. 2021. Interactive layout transfer. In *Proceedings of the 26th International Conference on Intelligent User Interfaces*, 70–80.

Deka, B.; Huang, Z.; Franzen, C.; Hibschman, J.; Afergan, D.; Li, Y.; Nichols, J.; and Kumar, R. 2017. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th annual ACM symposium on user interface software and technology*, 845–854.

Fan, L.; Wang, G.; Jiang, Y.; Mandlekar, A.; Yang, Y.; Zhu, H.; Tang, A.; Huang, D.-A.; Zhu, Y.; and Anandkumar, A. 2022. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, 35: 18343–18362.

Framer. 2025. Framer: Create a professional website, free. No code website builder loved by designers. https://www.framer.com. Accessed: 2025-11-27.

Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; and Dauphin, Y. N. 2017. Convolutional sequence to sequence learning. In *International conference on machine learning*, 1243–1252. PMLR.

Gonçalves, D.; Piçarra, M.; Pais, P.; Guerreiro, J.; and Rodrigues, A. 2023. " My Zelda Cane": Strategies Used by Blind Players to Play Visual-Centric Digital Games. In *Proceedings of the 2023 CHI conference on human factors in computing systems*, 1–15.

Guerreiro, J. J. A.; Inoue, N.; Masui, K.; Otani, M.; and Nakayama, H. 2024. Layoutflow: flow matching for layout generation. In *European Conference on Computer Vision*, 56–72. Springer.

Gupta, K.; Lazarow, J.; Achille, A.; Davis, L. S.; Mahadevan, V.; and Shrivastava, A. 2021. Layouttransformer: Layout generation and completion with self-attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1004–1014.

He, Z.; Sunkara, S.; Zang, X.; Xu, Y.; Liu, L.; Wichers, N.; Schubiner, G.; Lee, R.; and Chen, J. 2021. Actionbert: Leveraging user actions for semantic understanding of user interfaces. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 5931–5938.

Inoue, N.; Kikuchi, K.; Simo-Serra, E.; Otani, M.; and Yamaguchi, K. 2023. Layoutdm: Discrete diffusion model for controllable layout generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10167–10176.

Jiang, Y.; Zhou, C.; Garg, V.; and Oulasvirta, A. 2024. Graph4GUI: Graph Neural Networks for Representing Graphical User Interfaces. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 1–18.

Kumar, R.; Talton, J. O.; Ahmad, S.; and Klemmer, S. R. 2011a. Bricolage: example-based retargeting for web design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2197–2206.

Kumar, R.; Talton, J. O.; Ahmad, S.; Roughgarden, T.; and Klemmer, S. R. 2011b. Flexible tree matching. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence-Volume Volume Three*, 2674–2679.

Leiva, L. A.; Hota, A.; and Oulasvirta, A. 2020. Enrico: A dataset for topic modeling of mobile UI designs. In *22nd International Conference on Human-Computer Interaction with Mobile Devices and Services*, 1–4.

Li, G.; Baechler, G.; Tragut, M.; and Li, Y. 2022. Learning to denoise raw mobile UI layouts for improving datasets at scale. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, 1–13.

Li, Q.; Zeng, H.; Peng, Z.; and Ma, X. 2021a. Understanding players' interaction patterns with mobile game app UI via visualizations. In *Proceedings of the Ninth International Symposium of Chinese CHI*, 9–21.

Li, T. J.-J.; Popowski, L.; Mitchell, T.; and Myers, B. A. 2021b. Screen2vec: Semantic embedding of gui screens and gui components. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 1–15.

Liu, H.; Zhu, Z.; Becherini, G.; Peng, Y.; Su, M.; Zhou, Y.; Iwamoto, N.; Zheng, B.; and Black, M. J. 2023. Emage: Towards unified holistic co-speech gesture generation via masked audio gesture modeling. *arXiv preprint arXiv:2401.00374*.

Otani, M.; Inoue, N.; Kikuchi, K.; and Togashi, R. 2024. LTSim: Layout Transportation-based Similarity Measure for Evaluating Layout Generation. *arXiv preprint arXiv:2407.12356*.

Panayiotou, K.; Doumanidis, C.; Tsardoulias, E.; and Symeonidis, A. L. 2024. SmAuto: A domain-specific-language

for application development in smart environments. *Pervasive and Mobile Computing*, 101: 101931.

Parr, T. 2013. *The definitive ANTLR 4 reference*. Pragmatic Bookshelf.

Patil, A. G.; Li, M.; Fisher, M.; Savva, M.; and Zhang, H. 2021. Layoutgmn: Neural graph matching for structural layout similarity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11048–11057.

Perron, L.; and Furnon, V. 2024. Route. Schedule. Plan. Assign. Pack. Solve. https://developers.google.com/optimization/. Accessed: 2025-11-27.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Su, J.; Ahmed, M.; Lu, Y.; Pan, S.; Bo, W.; and Liu, Y. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568: 127063.

Takada, A.; Yamazaki, D.; Yoshida, Y.; Ganbat, N.; Shimotomai, T.; Hamada, N.; Liu, L.; Yamamoto, T.; and Sakurai, D. 2023. Genélive! generating rhythm actions in love live! In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 5266–5275.

Uizard. 2024. Uizard: UI Design Made Easy, Powered By AI. https://uizard.io. Accessed: 2025-11-27.

Vaswani, A. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *nature*, 575(7782): 350–354.

Warner, J.; Kim, K. W.; and Hartmann, B. 2023. Interactive Flexible Style Transfer for Vector Graphics. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 1–14.

Wu, J.; Swearngin, A.; Zhang, X.; Nichols, J.; and Bigham, J. P. 2023. Screen correspondence: Mapping interchangeable elements between uis. *arXiv preprint arXiv:2301.08372*.

Wu, J.; Zhang, X.; Nichols, J.; and Bigham, J. P. 2021. Screen parsing: Towards reverse engineering of ui models from screenshots. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, 470–483.

Xu, P.; Li, Y.; Yang, Z.; Shi, W.; Fu, H.; and Huang, H. 2022. Hierarchical layout blending with recursive optimal correspondence. *ACM Transactions on Graphics (TOG)*, 41(6): 1–15.

Yamaguchi, K. 2021. Canvasvae: Learning to generate vector graphic documents. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5481–5489.

Ye, D.; Liu, Z.; Sun, M.; Shi, B.; Zhao, P.; Wu, H.; Yu, H.; Yang, S.; Wu, X.; Guo, Q.; et al. 2020. Mastering complex control in moba games with deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 6672–6679.

Zhao, Z.; Lai, Z.; Lin, Q.; Zhao, Y.; Liu, H.; Yang, S.; Feng, Y.; Yang, M.; Zhang, S.; Yang, X.; et al. 2025. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation. *arXiv preprint arXiv:2501.12202*.