Hack More: Hackathon on Modelling and Rendering

F. Ganovelli [1] N. Capece[2] F. Ponchio[1] K. Lupinetti[3] and M. Tarini[4]

[1]Visual Computing Lab - ISTI-CNR, Italy
[2]Gaphics 3D Lab - Unibas, Italy
[3] IMATI-CNR, Genova
[4] Unimi, Italy

Abstract

Hack More was a one-day educational hackathon open to all and designed to be accessible to participants with any level of background in Computer Graphics, with a particular focus on bachelor-level students and with the goal of attracting young talents to the field. The challenge centered on modeling and rendering through Constructive Solid Geometry (CSG), a topic rarely emphasized in standard Computer Graphics courses. This choice helped ensure a more uniform baseline among participants, required minimal prerequisites, and offered an engaging and stimulating learning experience. In this paper, we describe the motivation and the organization leading to the event, and the material and tools that were developed. Finally, we discuss opportunities and directions for future hackathons, based on the feedback collected from this experience.

CCS Concepts

• ***Computing methodologies*** → ***Mesh geometry models; Ray tracing;*** *Solid Modelling;* • ***Applied computing*** → ***Interactive learning environments; Computer-assisted instruction;***

1. Introduction and context

In the last decade, education in Computer Graphics (CG) and, more generally, in Computer Science has faced notable pedagogical challenges. Educators struggle to engage students through conventional formats of lectures followed by laboratory exercises, as the passive presentation of material often fails to motivate students or convey the practical depth of CG topics. Studies indicate that replacing purely lecture-based teaching with interactive, project-based learning strategies can significantly improve student engagement and learning outcomes in CG education, although such approaches may require greater effort from both students and teachers [RMV*21, Lia17]. These findings further support the effectiveness of student-centered and experiential learning approaches, highlighting how project-based activities can bridge the gap between theoretical education and the development of practical skills relevant to real-world applications [AKAZ24]. Modern CG courses are increasingly adopting active, project-based methods to make core concepts tangible and to support student motivation.

In parallel, among pedagogical tools aimed at promoting innovation, experiential learning, and the practical application of theoretical knowledge, hackathons have gained increasing relevance [CGHA]. Originally conceived as time-constrained programming competitions, hackathons are now defined as intensive collaborative events where small teams work under pressure to prototype solutions to specific challenges. Participants spend from a few hours to days in focused problem-solving, which both tests and deepens their knowledge of the underlying concepts [AKB25]. By engaging participants in intensive, time-constrained collaborative problem-solving activities, hackathons provide experiential learning environments that simulate real-world conditions and effectively support the integration of theoretical knowledge with practical skills, while promoting teamwork, creativity, and technical expertise [SGBR25].

With this premise, we conceived *Hack More*, an educational hackathon focused on CG. It was designed and organized by research scientists in the CG field, belonging to the Hack More Association [ANO], whose mission includes promoting greater awareness of and interest in CG in Hack More, as well as improving communication among individuals with an interest in CG. Therefore, the hackathon fits naturally within the range of activities carried out by Hack More, which already includes the organization of conferences, discussion groups, and workshops, and the maintenance of relationships with other similar national and international associations. By integrating the hackathon into Hack More portfolio, we leveraged the association's network and communication channels to publicize the event and strengthen its educational mission.

From an awareness-raising perspective, Hack More specifically aimed to connect students and individuals with limited or no prior research experience closer to the field of CG. In particular, our target participants included undergraduate students in bachelor's de-

DOI: 10.2312/eged.20261002

gree programs who may have learned some programming or graphics basics but had not engaged in academic research. The hackathon challenge was designed to be accessible yet substantial, so that teams could learn by doing, without excessive prior knowledge. To encourage broad participation and sustained effort, we offered a monetary prize and an official award from the Hack More to the team that best faced the challenge. The winning team received a 500-euro prize and a 3D-printed Hack More logo with a wood-effect mounted on a transparent frame, providing both incentive and recognition within the learning experience.

The event structure encouraged students to work in small teams, deliberately composed of complementary skills, pairing a 3D modeler/designer with a developer. This configuration mirrors real-world CG projects, which require both aesthetic and technical expertise, and allows teammates to learn from each other's strengths. By providing only a basic set of guidelines and focusing the challenge around the theme *Synthetic Yet Organic: Crafting Life from Primitives*, Hack More encouraged teams to rapidly prototype solutions under time constraints, experiencing a condensed and open-ended CG development cycle, from conceptual exploration to the presentation of a tangible result, that goes beyond what is usually achievable through traditional lectures alone.

2. Related Work

Hackathons have been increasingly studied as educational and collaborative formats within Computer Science and Software Engineering [CG23, HSRM22, PKI*18]. Prior work characterizes hackathons as short, intensive, and team-based events that promote rapid prototyping and experiential engagement with complex technical challenges [NM16]. Several studies report positive effects on students' motivation, engagement, and the acquisition of practical skills, particularly when traditional lecture-based teaching is insufficient to fully convey the complexity of the topics covered in teaching activities. Within computer science, hackathons have been adopted as informal or semi-formal learning environments, often supplementing standard curricula by emphasizing collaboration, creativity, and autonomy [PKI*19]. Research has highlighted their effectiveness in promoting problem-solving skills and interdisciplinary thinking, while highlighting the challenges related to assessment, time constraints, and participants' diverse backgrounds [SC24]. As shown by [Edm25], these events provide a risk-free milestone where students can experience a full development cycle and exercise the opportunity to fail without academic penalty. However, existing studies focus on general programming, software engineering, data science, or hackathon design principles rather than the educational particularities of a specific domain. Consequently, there is limited empirical evidence and guidance on how to adapt hackathons to domains whose competence skill profiles and assessment criteria differ substantially from traditional programming activities.

CG is one such domain. Teaching CG, modeling, and rendering are topics widely recognized as particularly challenging due to the combination of a strong mathematical foundation (linear algebra, geometry, numerical methods), with diverse algorithmic reasoning (visibility, shading, sampling), a broad range of software tools (APIs, rendering systems, and visualization pipelines), and significant design and implementation aspects. Prior work has shown that effective CG education requires curricula that integrate both theoretical principles and practical laboratory experiences, including low and high-level algorithms, geometric transformations, visualization pipelines, and application programming interfaces, and that these components should be tailored to the needs of different engineering and computer science programs rather than treated as isolated topics [Paq05, dS01]. Prior educational approaches in CG emphasize incremental assignments and visual feedback to support conceptual understanding and practical competence [Rom13]. In this context, activity-led learning has been proposed to introduce the subject in a broad and accessible way, allowing students to understand the overall field of CG before focusing on complex details. For example, [APH*12] showed that involving students in a six-week challenge, such as building a 3D drawing application with a hardware interface, can reduce negative perceptions about computing degrees and improve student retention through engaging, project-based learning. However, the intensive and creative nature of graphics pipelines, particularly in modeling, rendering, and geometric processing, poses unique educational challenges that are not fully addressed by typical course structures. Unlike many programming tasks, where correctness can be judged objectively through a series of test suites, success in CG depends on a combination of objective criteria (*e.g.*, performance, numerical stability, correctness of geometric operations) and subjective criteria (visual quality, aesthetics, perceptual plausibility).

CG projects, therefore, require: *(i)* knowledge or the ability to learn specialized software tools; *(ii)* careful integration of visualization debugging workflow; *(iii)* time for iterative tuning of modelling parameters and rendering; *(iv)* evaluation protocols that combine quantitative with qualitative review. However, recent studies in computer science education emphasize conventional or project-based learning frameworks, such as systematic investigation of how project-based learning is implemented and assessed in software-related courses [SHL*25] or analyses of classroom organization and teaching methods in higher education [Sha25]. These contributions offer valuable insights into organizing learning activities and assessing practical outcomes, but they do not specifically address the educational needs of CG, where domain-specific knowledge, visual and geometric reasoning, and the integration of modeling and rendering tools introduce distinctive educational challenges that go beyond standard software engineering or classical project-oriented paradigms. In this context, activity-led instruction highlights the importance of integrative tasks, which enable students to combine different skills, such as mathematical knowledge and creative design, while solving real-world problems. This supports a shift from traditional lectures to student-centered, learning by doing approaches, which are essential for preparing graduates for the creative industries [AP09]. Our approach also aligns with broader efforts to integrate research and creative production in CG education. In this context, [GMP14] describes a pedagogical method that groups individual student research topics into a collective artistic project, simulating professional studio pipelines to solve specific technical and aesthetic stakes. Similarly, [AMG*24] emphasizes the importance of promoting undergraduate research through extracurricular and practice-based initiatives that emphasize the opportunity to fail. [SWLR17] suggests that a top-down approach,

starting with a complex project like a game or simulation and breaking it down into modules, can increase student motivation and help them to absorb the foundations of the field through practice. However, such project-based approaches often require high levels of supervision and can be difficult to implement in large-scale settings. These strategies, much like Hack More, aim to bridge the gap between vocational training and the advanced research and development skills demanded by the creative industries by providing a low-risk environment for experimentation and independent problem-solving.

Despite the growing adoption of hackathons in Computer Science education, there is a notable lack of studies examining hackathon-based learning experiences explicitly designed for CG and visual computing. To the best of our knowledge, no prior work systematically investigates hackathons centered on modeling and rendering tasks, nor on geometric representations such as constructive solid geometry, which require both conceptual rigor and hands-on experimentation. Our work is positioned at the intersection of these research areas by presenting and analyzing Hack More, a hackathon specifically designed for modeling and rendering in CG. By focusing on domain-specific challenges, tools, and learning objectives, we aim to contribute empirical insights into the effectiveness and limitations of hackathon-based formats for CG education, complementing the existing literature on both hackathons and CG education. Finally, Hack More try to blend programming with artistic experimentation by encouraging participants to use their imagination to face the difficulties of converting strict technical and platform constraints into creative signatures [FE26].

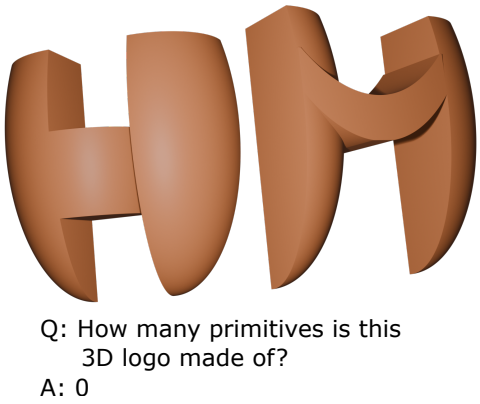

Figure 1: *The image sent to the participants two days before the hackathon to hint what the challenge could be about.*

3. The challenge

The proposed challenge consisted of modelling a scene with Constructive Solid Geometry (CSG) [Fol96] and writing the code to render it. This choice is motivated by a number of considerations.

Similar starting conditions. CSG is a topic typically referred in the typical introductory CG courses, but not investigated in depth, which means that we could be reasonably confident that all the participants could start from a similar level of knowledge.

Ability in modelling, not accomplished users. We wanted the modelers to be able to demonstrate their creativity as 3D artists without relying primarily on their proficiency with specific software suites such as Blender. To our knowledge, none of the major industrial 3D modeling tools is designed around the CSG paradigm, even though it can be approximated. Instead, we suggested the use of OpenSCAD [Mar24], an open-source, lightweight application that operates through a simple scripting language.

Minimum background knowledge for feasibility and simplicity. CSG concepts are intuitive and easy to grasp, and they can be explained in a matter of minutes. This is a good fit, given the duration of our event (10.5 hours).

We designed the challenge to be completed in teams of two participants: one acting as the **modeler** and the other as the rendering **developer**. The modeler is tasked with the creation of the 3D object or scene, while the developer is expected to implement the rendering engine. This structure implies a balance and coordination between the two roles. It would be pointless to author a scene if the renderer is unable to handle it properly (in terms of scalability, support of operations and primitives, etc); viceversa, supporting more features in the renderer is only useful if the authored scene makes (good) use of them. Ultimately, the more the two team members complemented each other's contributions, the better the final result.

3.1. Introduction to the challenge

The precise nature of the challenge was not revealed to participants in advance. All that was publicly announced is the nature of the teams (modeler, developers), which could be registered as such or formed on the spot. As a teaser, emails were sent to participants two days before the hackathon with the hint displayed in Figure 1. Upon arrival, the participants were presented with an introductory talk (one hour) explaining the task and the basic nature of the algorithms to complete it. This talk covered the following topics:

- definition of CSG: union, intersection, and difference operators between a pair of solids and CSG tree;
- demo of OpenSCAD as a modelling tool;
- basics of ray-tracing: definition, building the rays, ray-primitive intersections, normal computation;
- direct ray tracing the CSG tree, ray intersection as a list of spans and operators of pair of lists;
- ray marching Signed Distance Function (SDF): definition of SDF, ray marching techniques;
- practical setup and file formats (see Section 4).

We allowed the use of automated tools such as large language models, but only in their free versions, to avoid giving an unfair advantage to participants with paid subscriptions. We considered prohibiting or restricting the use of AI, but we concluded that doing so would have been both impractical and out of touch with current practices. The implications of allowing these tools are discussed in Section 7.

4. An easy setup

We wanted participants to direct their effort as much as possible to the CG challenge itself, and not to set up the work environment, or on preliminary, tedious programming tasks. At the same time, we wanted to guarantee full freedom in the choice of programming languages, IDE and platform (as clearly notified to the participants, two weeks before the event, at the time of registration). These conflicting requirements dictated a few choices, explained in the following.

First of all, we limited the required software installation to OpenSCAD [Mar24]. OpenSCAD offers a solid one-click multiplatform installer (running on Windows, Linux, and Mac). A crucial choice concerned the interchange format defining the authored CSG objects, which serve as the common ground connecting the participants of the two categories (modelers and rendering developers).

OpenSCAD uses a full scripting language (.scad) to describe 3D models and scenes, supporting variables, modules, loops, functions, and parametric design. However, we needed to avoid burdening developers with the need to write a parser for this language, supporting all its many features. However, OpenSCAD can export in a format named .csg, which is still human-readable, but no longer a scripting language. Instead, the .csg format records the evaluated CSG tree directly, as a flat, explicit sequence of primitives, transformations, and Boolean operations with all variables resolved and all loops unrolled. The tree encoded with the CSG format has three types of nodes: an actual ($\cup \cap \setminus$) operation, a color, or a transformation matrix, the latter two meant to be applied to the sub-tree rooted at the node.

While the CSG format is much easier to parse, but still disproportionate to its importance in our context. Instead, we employed our own custom format, specifically designed for trivial parsing in any language, without compromising on the expressiveness. Our format is obtained from the CSG by visiting the tree and cumulating color and transformation operation down to the leaves, so that we obtain a tree where the nodes are only operators and where each leaf is a primitive with a matrix transformation and a color.

We made a further simplification: in the CSG specification, solid primitives defined on the leaf nodes (of type *sphere*, *cube*, or *cylinder*) can be centered either at the origin or on a custom point, and their extension is explicitly stated; in our simplified format, primitives are always implicitly unit-sized and origin-centered, their original dimension and position being simply embedded in the transformation matrix associated to the leaf.

Finally, our encoded CSG tree is always binary: for example, the union of three objects is represented by two unions in cascade. Our custom format is based on JSON, whose parsing is supported by either libraries or core features in most programming languages. Figure 3 exemplifies our custom specification for the object shown in Figure 2.

Participants could convert their own solid objects to our simplified format by means of a LAN-based web service, by uploading the CSG file exported by OpenSCAD, and immediately downloading the converted description. By providing this service as a web-based online tool, we avoided the need of installation of any additional tool.

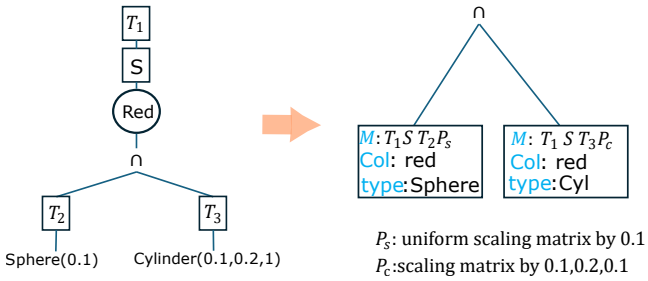

Figure 2: *Left: an example of the tree encoded by the CSG format. The nodes can be CSG operators or color and transformations affecting the whole subtree. Right: the corresponding CSG tree with transformation and color stored on the leaves/primitives.*

```
{
  "primitives": [
    {
      "matrix": [
        [1.000000, 0.000000, 0.000000, 0.000000],
        [0.000000, 1.000000, 0.000000, 0.000000],
        [0.000000, 0.000000, 1.000000, 0.000000],
        [0.000000, 0.000000, 0.000000, 1.000000]
      ],
      "color": [0.000000, 1.000000, 0.000000, 1.000000],
      "type": 0,   // 0-sphere, 1-cube, 2-cylinder
      "id_node": 1 // reference to the corresponding leaf in nodes
    },
    {
      "matrix": [
        [1.100000, 0.000000, 0.000000, 0.000000],
        [0.000000, 1.100000, 0.000000, 0.000000],
        [0.000000, 0.000000, 1.100000, 0.000000],
        [0.000000, 0.000000, 0.000000, 1.000000]
      ],
      "color": [0.000000, 1.000000, 0.000000, 1.000000],
      "type": 2,
      "id_node": 2
    }
  ],
  "nodes": [-1, 0, 1], // -1: union, -2: intersection, -3:
       difference, >0: reference to the primitive
  "parent": [-1, 0, 0],
  "children": [[1, 2], null, null]
}
```

Figure 3: *Encoding of the CSG tree. Additional comments have been added for clarity, and are not produced by our tools.*

5. Suggested path

Our target participants include students who only attended introductory university courses on CG; therefore, we illustrated in the initial talk a 9-step suggested path, guiding them from scratch to a first working prototype (see Figure 4):

1. Viewing ray setup. Write the code to build one ray per pixel.
2. Ray-sphere intersection. Render one sphere at the center of the view, compute the intersection point and normal.
3. Add some simple lighting and shading (necessary to produce readable images).

4. Write code to load test_00 (a sphere) and test_01 (a sphere non-uniformly scaled, rotated, and translated); this is a test to check if the transformation matrix is correctly interpreted and used.
5. Implement the union of spheres (test_02, test_03, test_04). This is the first CSG operator on 3 objects made just with spheres.
6. Implement the intersection of spheres (test_05,test_06)
7. Implement the difference between spheres (test_07)
8. Rendering a full CSG (test_08). This model combines 16 spherical primitives using all boolean operations.
9. As the last step: add different primitive types: cubes and cylinders. The proposed tests for this step were the model displayed on the Wikipedia page for CSG, and the model we used to 3D print the logo of the competition.

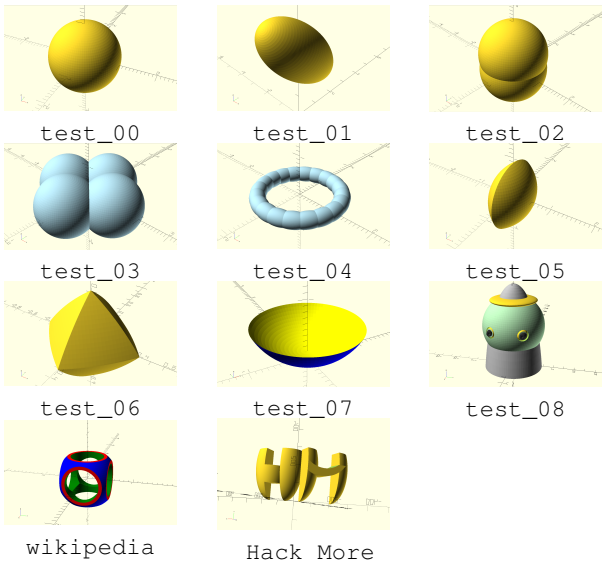

Figure 4: *The models given to the participants to test their code.*

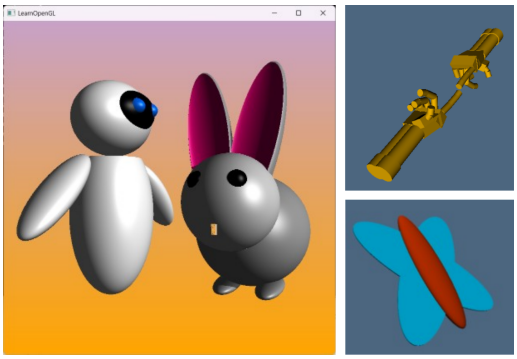

Figure 5: *Some rendering results.*

Clarity and organization of the submitted code were also considered, favoring implementations that were well-structured and readable. Although computational efficiency was not a formal requirement, noticeable optimizations that improved rendering time without compromising correctness were acknowledged informally.

Table 1: *Summary of evaluation criteria and weights.*

Criterion	Description	Weight
Final rendering outcome	Most advanced working stage; correctness and visual consistency	60%
CSG-based modelling	Clarity of structure; use of transformations; creativity	25%
Code clarity	Readability, structure, maintainability	15%
Efficiency (bonus)	Simple optimizations without loss of correctness	+5%

6. Evaluation

A jury evaluated each submission by participants and assigned prizes. The evaluation focused on the correctness and completeness of each team's final outcome within the progressive structure of the challenge. Since every step in the development path was strictly propaedeutical to the next, the assessment concentrated on the most advanced stage reached.

Projects were primarily judged on their demonstrated ability to generate visually coherent renderings for the reference scenes, with particular attention to the correct handling of hierarchical transformations, CSG operations (union, intersection, difference), and basic shading. The final outputs were compared with the expected results for the provided test suite.

The modelling assignment, created in OpenSCAD, was evaluated based on the meaningful use of CSG concepts, structural clarity, effective application of transformations, and creativity. The objective was to assess both conceptual understanding of CSG modelling and artistic talent.

7. Discussion

This section discusses the main outcomes of the hackathon, based on direct observation during the event and examination of the submitted codebases, as well as a questionnaire administered after the hackathon. The questionnaire cannot be used for statistical analysis, given the low number of participants, but it provides a structured and reproducible account of their experience. Nine students participated in the event, divided into six teams: three pairs and three individual participants. Please note that the participants were free to form a team of two, as suggested, or participate by themselves. Only two participants considered themselves CG experts, while the remaining ranged between beginner and intermediate levels. Almost the entirety of the participants rated themselves between beginner and intermediate in ray tracing, Constructive Solid Geometry, and 3D modelling in general. None of the participants had previous knowledge of OpenSCAD.

All teams were able to produce a partially or fully working implementation within the intended time. The degree of completeness varied: some renderers lacked support for some of the prim-

itive types or boolean operations; others exhibited rendering bugs. However, every team reached the first functional milestone. Notably, the first rendered images appeared within approximately the first hour of the activity, indicating that the initial entry barrier was sufficiently low, as intended.

The rapid initial progress validates the premise of our work: the combination of a constrained problem domain (CSG), a minimal set of primitives, and a guided development path was effective in enabling participants to move quickly from a blank project to a visible result. Even incomplete implementations typically demonstrated a correct understanding of ray construction, basic intersection logic, and shading, confirming that the core learning objectives were attainable within the time frame.

The primary difficulty reportedly encountered by participants was debugging. This challenge stemmed not from algorithmic complexity alone, but also from the need to understand, adapt, and correct code that was not entirely written by the participants themselves. In many cases, parts of the codebase were generated with external assistance or adapted from existing sources, increasing the cognitive load required to reason about failures.

As seen in Section 5, we anticipated the importance of debugging and provided a structured development path accompanied by incremental test scenes, each isolating a specific feature (e.g., transformations, individual CSG operations). This approach proved helpful in identifying localized issues and validating intermediate steps. However, its effectiveness was inherently limited once multiple interacting components were combined, particularly in full CSG trees, where errors could originate from several layers of the pipeline. As a result, debugging complex scenes remained the dominant bottleneck in the later stages of development.

AI-based coding tools played a central role in the development process. They were widely used to bootstrap the rendering infrastructure, including tasks such as window creation, graphics library initialization, and basic application scaffolding. In several cases, the generated code clearly reflected that participants had provided the AI with the introductory slides and the assignment description, which was evident from consistent naming conventions and structural similarities across different submissions.

The effectiveness of AI assistance was further amplified by the availability of high-quality open-source resources and technical blogs on ray tracing, signed distance functions, and CSG modeling. As a result, AI-generated code was often technically sound at a local level. However, subtle mismatches in conventions—such as assumptions about primitive parameterization (e.g., whether a cube is centered at the origin or defined from a corner), coordinate systems, or normal orientation, frequently led to rendering artifacts or logical errors and required careful inspection and debugging.

Overall, AI tools significantly accelerated early development but did not eliminate the need for a solid understanding of the underlying geometric and rendering concepts, particularly when diagnosing non-obvious bugs.

7.1. The benchmark participant

One of the organizers participated the challenge (although not in an official capacity). This person contributed solely to the logistics of the event but not in the technical preparation, and therefore had no prior knowledge of the challenge content, at par with the "real" participants. Their experience provided with some additional confirmation (if anecdotal) on the efficacy of our design choices. The difficulty curve resulted to be well calibrated, the progression of milestones closely following the intended path. The activity resulted engaging and enjoyable, even for an experienced participant.

8. Conclusions

Hack More shows exemplifies how hackathons can be effectively adapted to domain-specific challenges in CG education. By focusing on modeling and rendering with CSG, the event engaged participants with varying levels of experience, providing a common starting point while introducing them to non-trivial geometric and algorithmic tasks. The combination of minimal initial guidance, a clear progressive development path, and complementary team roles encouraged collaboration, creativity, and iterative problem-solving. Most teams were able to produce functional implementations, with some completing the full modeling and rendering pipeline, showing that even a short, intensive format can support meaningful experiential learning and bridge theory with practice. At the same time, the limited duration constrained the depth of exploration, and not all participants were able to implement every aspect of the challenge. Hack More highlights the potential of hackathon-based approaches to complement traditional CG education, foster motivation and engagement, and provide a practical experience with realistic workflows in visual computing.

References

[AKAZ24] ALSMADI H., KANDASAMY G., AL KAFRI A., ZAHIRAH K. F.: Empowering computing students through multidisciplinary project based learning (pbl): Creating meaningful differences in the real world. *Social Sciences & Humanities Open 10* (2024), 101180. URL: https://www.sciencedirect.com/science/article/pii/S2590291124003772, doi:https://doi.org/10.1016/j.ssaho.2024.101180. 1

[AKB25] ARAUJO A. A., KALINOWSKI M., BALDASSARRE M. T.: Embracing experiential learning: Hackathons as an educational strategy for shaping soft skills in software engineering. In *2025 IEEE/ACM 37th International Conference on Software Engineering Education and Training (CSEE & T)* (Los Alamitos, CA, USA, May 2025), IEEE Computer Society, pp. 319–324. URL: https://doi.ieeecomputersociety.org/10.1109/CSEET66350.2025.00040, doi:10.1109/CSEET66350.2025.00040. 1

[AMG*24] ANDERSON E. F., MCLOUGHLIN L., GINGRICH O., KANELLOS E., ADZHIEV V.: Approaches to Nurturing Undergraduate Research in the Creative Industries - a UK Multi-Institutional Exploration. In *Eurographics 2024 - Education Papers* (2024), Sousa Santos B., Anderson E., (Eds.), The Eurographics Association. doi:10.2312/eged.20241005. 2

[ANO] ANONYMIZED: Anonymized association. 1

[AP09] ANDERSON E. F., PETERS C. E.: On the provision of a comprehensive computer graphics education in the context of computer games: An activity-led instruction approach. In *Eurographics 2009* (2009), Eurographics Association, pp. 7–14. 2

[APH*12] ANDERSON E. F., PETERS C. E., HALLORAN J., EVERY P., SHUTTLEWORTH J., LIAROKAPIS F., LANE R., RICHARDS M.: In at the deep end: An activity-led introduction to first year creative computing. 1852–1866. URL: https://doi.org/10.1111/j.

1467-8659.2012.03066.x, doi:10.1111/j.1467-8659.2012.03066.x. 2

[CG23] CHAU C. W., GERBER E. M.: On hackathons: A multidisciplinary literature review. CHI '23, Association for Computing Machinery. URL: https://doi.org/10.1145/3544548.3581234, doi:10.1145/3544548.3581234. 2

[CGHA] COSTA G., GARCÍA-HOLGADO A., ALVAREZ P.: Systematic mapping of the literature on hackathons. In *Frontiers in Education*, vol. 10, Frontiers, p. 1701443. 1

[dS01] DOS SANTOS M. P.: Computer graphics in the scope of informatics engineering education. *Computers & Graphics 25*, 5 (2001), 909–915. Mixed realities - beyond conventions. URL: https://www.sciencedirect.com/science/article/pii/S0097849301001315, doi:https://doi.org/10.1016/S0097-8493(01)00131-5. 2

[Edm25] EDMOND C.: Extra lives: game jam as extra-curricular learning for university students. In *Proceedings of DiGRA Australia 2025* (2025), Digital Games Research Association (DiGRA). DiGRAA 2025 ; Conference date: 05-02-2025 Through 07-02-2025. URL: https://digraa.org/post/category/digraa2025/. 2

[FE26] FURMAN I., ERDIKMEN A.: The demoscene: from digital subculture to unesco intangible cultural heritage. *International Journal of Heritage Studies 32*, 3 (2026), 418–434. doi:10.1080/13527258.2025.2595079. 3

[Fol96] FOLEY J. D.: *Computer graphics: principles and practice*, vol. 12110. Addison-Wesley Professional, 1996. 3

[GMP14] GEORGE-MOLLAND A.-L., PLESSIET C.: Producing Creative Artistic Projects by Grouping Students' Computer Graphics Research Topics. In *Eurographics 2014 - Education Papers* (2014), Bourdin J.-J., Jorge J., Anderson E., (Eds.), The Eurographics Association. doi:10.2312/eged.20141031. 2

[HSRM22] HAQUE R., SALMANI A., RAESSINEJAD N., MOSHIRPOUR M.: Effectiveness of hackathons in software engineering education. In *2022 IEEE Frontiers in Education Conference (FIE)* (2022), pp. 1–8. doi:10.1109/FIE56618.2022.9962527. 2

[Lia17] LIAROKAPIS F.: Using Activity Led Learning for Teaching Computer Graphics Principles Through Augmented Reality. In *EG 2017 - Education Papers* (2017), Bourdin J.-J., Shesh A., (Eds.), The Eurographics Association. doi:10.2312/eged.20171025. 1

[Mar24] MARIUS KINTEL: Openscad. https://openscad.org, 2024. Version 2024.01, accessed 2026-02-05. 3, 4

[NM16] NANDI A., MANDERNACH M.: Hackathons as an informal learning platform. SIGCSE '16, Association for Computing Machinery, p. 346–351. URL: https://doi.org/10.1145/2839509.2844590, doi:10.1145/2839509.2844590. 2

[Paq05] PAQUETTE E.: Computer graphics education in different curricula: analysis and proposal for courses. *Computers & Graphics 29*, 2 (2005), 245–255. URL: https://www.sciencedirect.com/science/article/pii/S0097849304002225, doi:https://doi.org/10.1016/j.cag.2004.12.011. 2

[PKI*18] PORRAS J., KHAKUREL J., IKONEN J., HAPPONEN A., KNUTAS A., HERALA A., DRÖGEHORN O.: Hackathons in software engineering education: lessons learned from a decade of events. SEEM '18, Association for Computing Machinery, p. 40–47. URL: https://doi.org/10.1145/3194779.3194783, doi:10.1145/3194779.3194783. 2

[PKI*19] PORRAS J., KNUTAS A., IKONEN J., HAPPONEN A., KHAKUREL J., HERALA A.: Code camps and hackathons in education - literature review and lessons learned. In *Hawaii International Conference on System Sciences* (2019). URL: https://api.semanticscholar.org/CorpusID:102352264. 2

[RMV*21] RODRIGUES R., MATOS T., VALLE DE CARVALHO A., BARBOSA J. G., ASSAF R., NÓBREGA R., COELHO A., DE SOUSA A. A.: Computer graphics teaching challenges: Guidelines for balancing depth, complexity and mentoring in a confinement context. *Graphics and Visual Computing 4* (2021), 200021. URL: https://www.sciencedirect.com/science/article/pii/S2666629421000048, doi:https://doi.org/10.1016/j.gvc.2021.200021. 1

[Rom13] ROMERO M.: Project-Based Learning of Advanced Computer Graphics and Interaction. In *Eurographics 2013 - Education Papers* (2013), Bourdin J.-J., Cerezo E., Cunningham S., (Eds.), The Eurographics Association. doi:10.2312/conf/EG2013/education/001-006. 2

[SC24] SCHULTEN C., CHOUNTA I.-A.: How do we learn in and from hackathons? a systematic literature review. *Education and Information Technologies* (2024). Accessed via web search. URL: https://link.springer.com/article/10.1007/s10639-024-12668-1. 2

[SGBR25] SOTAQUIRÁ-GUTIÉRREZ R., BELTRAN L. M., RUIZ J. P. G.: Hackathons as experiential learning platforms for engineering design skills. *Cogent Education 12*, 1 (2025), 2442187. URL: https://doi.org/10.1080/2331186X.2024.2442187, arXiv:https://doi.org/10.1080/2331186X.2024.2442187, doi:10.1080/2331186X.2024.2442187. 1

[Sha25] SHAVDIROV S.: Method of organization of classes in higher education institutions using flipped classroom technology. *AIP Conference Proceedings 3268*, 1 (02 2025), 070035. URL: https://doi.org/10.1063/5.0257187, arXiv:https://pubs.aip.org/aip/acp/article-pdf/doi/10.1063/5.0257187/20409147/070035_1_5.0257187.pdf, doi:10.1063/5.0257187. 2

[SHL*25] SULEIMAN A. D., HOU D., LIU Y., DEWATERS J., SHEPHERD D. C., G. DE SOUZA J.: Systematic literature review on project-based learning in computing education. *ACM Trans. Comput. Educ. 25*, 4 (Oct. 2025). URL: https://doi.org/10.1145/3743684, doi:10.1145/3743684. 2

[SWLR17] SUSELO T., WÜNSCHE B., LUXTON-REILLY A.: The journey to improve teaching computer graphics: A systematic review. In *International Conference on Computers in Education* (2017). 2