MATHEMATICS FOR GRAPHICS COMPUTING: students learn Algebra and program Python to create a project where they make Algebra create a scenario's photo.

Sandra Gaspar Martins[1,2], José Manuel Matos[2], João Beleza de Sousa[1], Lucía Suárez[1]

[1]*Instituto Superior de Engenharia de Lisboa- IPL, Portugal;* [2]*Faculdade de Ciências e Tecnologia da UNL – UIED, Portugal*

sandra.martins@adm.isel.pt, jmm@fct.unl.pt, jsousa@deetc.isel.ipl.pt, lsuarez@adm.isel.pt

This paper presents and evaluates an innovative multidisciplinary approach to teach algebra. Students learn, at the same course, algebra and object-oriented programming in Python (a programming language). Both contents are widely used to create a final project. The final project is a python program that, using the usual algebra sylabus simulates a camera; Given, as input, a 3D scene created using triangles (three 3Dpoints), some light sources (3Dpoints) and a point of view (3Dpoint) from where the "photo" is taken it gives, as output, an image file with the photo of that scene. Many students revealed that to create an immediate application of algebra enhances their engagement. Other input (other scene, other lights or other point of view) produces as output a new photo. Many of the participants: teachers, students and researchers, evaluate positively this multi contents course besides the hard work it demands from all.

Keywords: Algebra; Python; Informatics Undergraduate Students; Applications; Engagement

INTRODUCTION

The integration of mathematics and computers is spread all over research. For example, in the *Principles and Standards of School Mathematics the National Council of Teachers of Mathematics* (NCTM, 2000) one of six principles for high quality mathematics education is the "Technology principle". The act of programming a mathematical concept makes school practice not essentially repetitive and foster reflective teaching experiences (Teixeira, Matos, & Domingos, 2015). Michele Artigue (2016) argues that dramatic changes come with the technological evolution in the ways that teachers and students access information and resources, learn, communicate, interact, work and produce with others. Using programation to teach mathematics is certainly one of those dramatic changes.

In a review of research on project-based learning, J. W. Thomas (2000) found many studies involving project based learning and concluded that although it has some limitations, its evaluation was positive among students and teachers. Mills and Treagust (2003) states that "the use of project-based learning as a key component of engineering programs should be promulgated as widely as possible, because it is certainly clear that any improvement to the existing lecture-centric programs that dominate engineering would be welcomed by students, industry and accreditors alike."

To contrast programming languages, Fangohr (2004) made a comparison of C, MATLAB and Python as teaching languages for engineering students. His study comprised two phases: to make an algorithm to solve a problem and to translate it to a programming language. He found Python as the best choice in terms of clarity and functionality. Python was also used to teach mathematics (Schliep & Hochstättler, 2002) since teaching algorithms is one of the natural applications of multimedia in mathematics. The mathematics objects that they considered were of a highly dynamic nature and require an adequate dynamic visualization using Python. Students also prefered Python as the first programming language to learn when compared to Java and other commercial languages (Radenski & Atanas, 2006).

CONTEXT

In the semester of 2011/12, at Instituto Superior de Engenharia de Lisboa, Instituto Politécnico de Lisboa, was developed a new course, Multimedia and Computer Science Engineering Graduation, and a course of Algebra for graphic computation conceived by a mathematician (Carlos Leandro) and then itteratively improved by another mathematician (Lucía Suárez) both working together with a computer scientist (João Beleza de Sousa). The syllabus

was conceived from the beginning and all Algebra concepts taught were programmed in Python and used to create the project. This is a second semester curricular unit, students already know the programming basics, and are beginning object oriented programming. The course has weekly classes of 1h30m of Python and 3h of Algebra, all taught by the same professor (some are mathematicians, others are computer scientists). The classification is obtained 60% from exam, 20% from online homework and 20% from individual final project with required discussion (students must grade higher than 50% in every one of the 3 parts).

The course goal, as illustrated in the final project, is to develop a program in Python that when the user inserts (input) a 3D scenario made of triangles (three 3D points) with an associated RGB colour (triple/3D point), including the position (3D point) and colour of some lights (RGB colour-triple) and the position of the camera (3D point), the output is a photo of that scenario. If the input are different triangles, lights or position of the camera it immediately produces a different photo as a ".ppm" file.

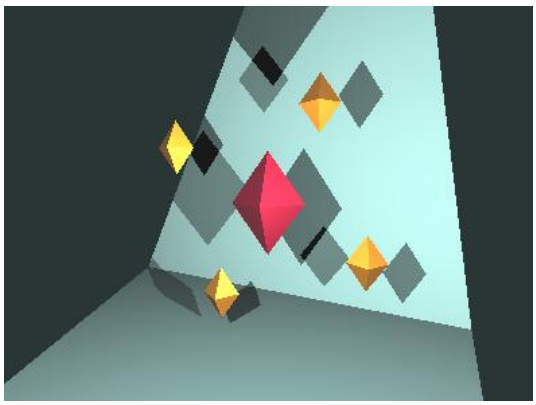

Figure 1. Example of the output of a student's project (not totally correct but very illustrative). The student made a scenario with a background big blue trinangle and five pyramids: four yellow pyramids and one red. Two light focuses give rise to two shadows for each pyramid and the different face colours dependent of the light incidence.

To create that program, students work with 3D points, vectors and matrices. The whole syllabus was designed to be nearly all applied in this project. The syllabus is:

1. Matrices operations; Inverse matrix; Determinants; Linear equation systems; Proper values and vectors.
2. 2D and 3D: Vectors and points; Referential and coordinates; Lines and plans; Internal and external product; Angles; Barycentric coordinates.
3. Geometric transformations (rotations, translations, scaling, …); Homogeneous coordinates and matricial representation; Perspective and parallel projections.
4. Surfaces: Intersection of lines and planes; normal vectors, reflection and refraction.

DETAILED PROJECT

In "Algebra" classes, which are theorethical/practical, students are taught traditional Algebra.

In "Python" classes, typically every week, students program a class/object into a file, by themselves with the natural support of teachers, using a guide provided before by teachers. Those students are Computer Science students and all take their own labtop to class, all have one – this is not an issue. The first week there is an example, out of context, to teach students the basics of object oriented programming, for example, students create a class named Circle with the data "radius", its "constructor" and some operations as: area, perimeter; double_perimeter (whose output is the double of the perimeter) and n_perimeter (analogous). In the following weeks, student program every class (part of the program) that are needed for the final project.

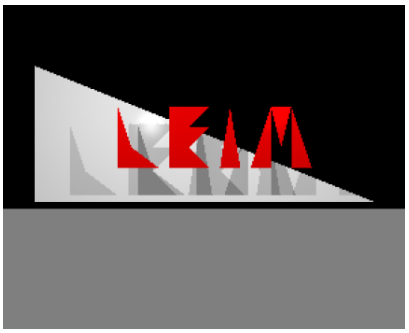

Figure 2 . Example of an image produced by a Ray Tracer project (as output)

For example, to produce the image in figure 2, the input is:

Colors

red = ColorRGB(1.0, 0.0, 0.0)

green = ColorRGB(1.0, 1.0, 1.0)

black = ColorRGB(0.0, 0.0, 0.0)

grey = ColorRGB(0.25, 0.25, 0.25)

bright = 100.0

Triangles

letter L - triangule 1

l1-v1 = Point3D(-4.25, 0.0, 0.0)

l1-v2 = Point3D(-3.25, 0.0, 0.0)

l1-v3 = Point3D(-4.25, 3.0, 0.0)

l1 = TriangleFace(l1-v1, l1-v2, l1-v3, letters-color)

letter L - triangule 2

l2-v1 = Point3D(-4.25, 0.0, 0.0)

l2-v2 = Point3D(-2.25, 0.0, 0.0)

l2-v3 = Point3D(-4.25, 1.5, 0.0)

l2 = TriangleFace(l2-v1, l2-v2, l2-v3, letters-color)

letter E - triangle 1

e1-v1 = Point3D(-1.75, 1.0, 0.0)

e1-v2 = Point3D(0.25, 3.0, 0.0)

e1-v3 = Point3D(-1.75, 3.0, 0.0)

e1 = TriangleFace(e1-v1, e1-v2, e1-v3, letters-color)

letter E - triangle 2

e2-v1 = Point3D(-1.75, 0.0, 0.0)

e2-v2 = Point3D(0.25, 2.0, 0.0)

e2-v3 = Point3D(-1.75, 2.0, 0.0)

e2 = TriangleFace(e2-v1, e2-v2, e2-v3, letters-color)

letter E - triangle 3

e3-v1 = Point3D(-1.75, 0.0, 0.0) e3-v2 = Point3D(0.25, 0.0, 0.0)

…

letter M - triangle 1

m3-v1 = Point3D(4.25, 0.0, 0.0)

m3-v2 = Point3D(5.25, 0.0, 0.0)

m3-v3 = Point3D(4.25, 3.0, 0.0)

m3 = TriangleFace(m3-v1, m3-v2, m3-v3, letters-color)

Background

background-v1 = Point3D(-10.0, -2.0, -2.0)

background-v2 = Point3D(10.0, -2.0, -2.0)

background-v3 = Point3D(-10.0, 6.0, -2.0)

background = TriangleFace(background-v1, background-v2, background-v3, background-color)

Floor

floor-v1 = Point3D(-12.0, -2.0, 0.0)

floor-v2 = Point3D(12.0, -2.0, 0.0)

floor-v3 = Point3D(0.0, -2.0, 1.0*10**4)

floor = TriangleFace(floor-v1, floor-v2, floor-v3, floor-color)

faces-list = [l1, l2, e1, e2, e3, i1, m1, m2, m3, background, floor]

Lights list

light1-position = Point3D(-5.0, 4.0, 5.0)

light2-position = Point3D(5.0, 4.0, 5.0)

light1 = PontualLight(light1-position,white, white ,white)

light2 = PontualLight(light2-position, white, white, white)

lights-list = [light1, light2]

The camera

camera-position = Point3D(0.0, 0.0, 10.0)

camera-looking-to = Point3D(0.0, 0.0, 0.0)

camera-vertical = Vetor3D(0.0, 1.0, 0.0)

camera-distance-eye-to-projection-plane= 5.0

camera-large-projection-rectangle = 10.0

camera-high-projection-rectangle = 8.0

camera-resolution-horizontal = 300

camera-vertical-resolution = 240

Main body

camera =. . .

background-color = black

ray-tracer = RayTracer(faces-list, lights-list, camera, background-color)

Roughly, we have a World Coordinates System where the scene (set of coloured triangles) is implemented; the Camera Coordinates System and a plane (projection plane) where the scene is projected and where the image is produced.

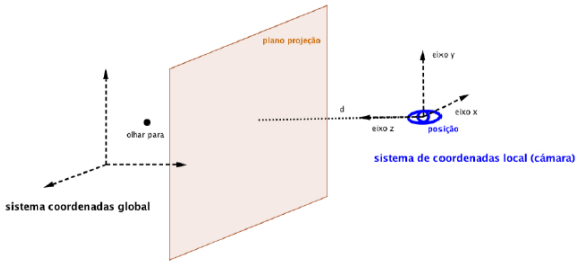

Figure 3. The two coordinate systems in use and the projection plan.

The "Ray Tracing" model will be explained now in detail. In order to produce an image from a 3D scene, a virtual camera is placed on the scene. The virtual camera projection surface is a rectangle placed at some distance from the camera position, along the view direction. This projection rectangle is divided into pixels. Since the image resolution is finite only a finite number of rays that came from each light must be followed.

Given that the virtual camera properties, such as the distance from the camera position to the projection rectangle, are specified in the Camera Coordinates System, and the scene, composed by triangles, is defined in the World Coordinates System, a simple ray tracing algorithm would be:

1) for each pixel in the projection rectangle create a ray (a line) that start at the camera position (the eye position-a 3D point) and passes through the pixel position (a 3D point).
2) convert the scene coordinates (3D points) from the World Coordinates System to the Camera Coordinates System (Referential change). This is because the ray constructed in the previous step (defined in the Camera Coordinates System) will be intercepted with the scene triangles defined in the World Coordinates System. This conversion is done using a matrix.
3) intercept each ray (line) with each triangle in the scene. Each triangle is defined by a plane equation. The interception of ray/triangle (equations system) is solved by the Crammer method (matrix determinants).
    a) if the ray does not intersect any triangle, use the scene background color as the pixel color.
    b) if the ray intercepts a set of triangles, choose the nearest one, as the visible one (this assumes that the triangles are opaque). Determining the nearest triangle is done by computing vectors (defined between the eye position and the interception) length.
        i) create new rays (lines) starting at the interception point and ending at each light source. If some other triangle in the scene intercepts this new ray, that means that the point is in shadow. In either case, in shadow or not, use a color model such as the Phong model, to get the color of the interception point.
        ii) add the contribution of all light sources to get the pixel color (sum of 3Dpoints).

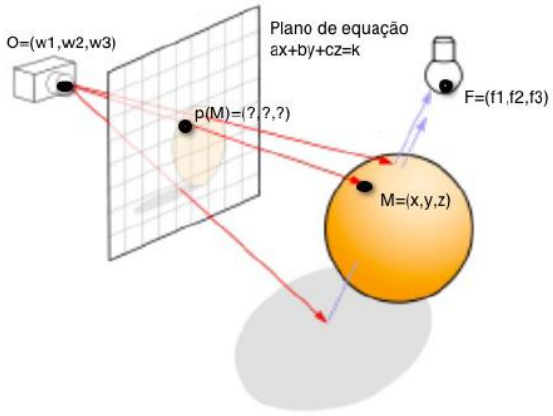

Figure 4. A scheme illustrating the Ray Tracer procedure.

The classes and operations created during the semester are below (names are in English and in Portuguese to be the

most natural possible).

- **CorRGB**(r,g,b): __init__; __repr__ ; soma ; __add__ ; set_hsv ; multiplica ; multiplica_escalar ;__mul__ .
- **Imagem**(numero_linhas, numero_colunas, linhas): __init__; __repr__ ;set_cor; get_cor; guardar_como_ppm.
- **Matriz**(numero_linhas, numero_colunas, linhas):__init__; __repr__; set_entrada; get_entrada; adiciona; __add__; transposta; multiplica; multiplica_escalar; __mul__; det_2x2; det_3x3; sub_matriz; det; copia; set_linha; set_coluna.
- **Ponto3D**(x,y,z):__init__; get_x; get_y; get_z; __repr__; adiciona_vetor; __add__; subtrai_ponto; __sub__.
- **LuzPontual**(posição; intensidade_ambiente; intensidade_difusa; intensidade_especular): __init__; __repr__; get_posicao; get_intensidade_ambiente; get_intensidade_difusa; get_intensidade_especular.
- **CorPhong**(k_ambiente; k_difusa; k_especular; brilho): __init__; __repr__; get_cor_rgb.
- **Reta**(origem, destino, vetor_diretor): __init__; __repr__; soma; __add__; set_hsv; multiplica; multiplica_escalar; __mul__.
- **Plano**(ponto1; ponto2; ponto3; normal): __init__; __repr__; interceta_triangulo.
- **FaceTriangular**(ponto1; ponto2; ponto3; normal; cor_phong): __init__; __repr__; get_cor_phong.
- **Camara**(posição; olhar_para; vertical; distancia_olho_plano_projecao; largura_retangulo_projecao ; altura_retangulo_projecao; resolucao_horizontal; resolucao_vertical; eixo_x; eixo_y; eixo_z; incremento_horizontal; incremento_vertical; canto_superior_esquerdo_x; canto_superior_esquerdo_y; canto_superior_esquerdo_z; matriz):

And the code of the final project is below, students must have all the previous classes completed and tested. And students should create the remaining methods, the scenario and test their program.

Code for ray_tracer_xxxxx.py

```
from ponto_xxxxx import Ponto3D
from cor_rgb_xxxxx import CorRGB
from cor_phong_xxxxx import CorPhong
from face_xxxxx import FaceTriangular
from luz_xxxxx import Luz
from vetor_xxxxx import Vetor3D
from camara_xxxxx import Camara
from reta_xxxxx import Reta
from imagem_xxxxx import Imagem
class RayTracer:
# miss constructeur
# miss the method __str__
# miss the method renderiza
# tests
if __name__ == "__main__":
# constructeur test
# constructeur test - cor da face
verde = CorRGB(0.0, 0.3, 0.0)
brilho = 100.0
cor = CorPhong(verde, verde, verde, brilho)
# constructeur test - face
p1 = Ponto3D(0.0, 0.0, 0.0)
p2 = Ponto3D(1.0, 0.0, 0.0)
p3 = Ponto3D(0.0, 1.0, 0.0)
face = FaceTriangular(p1, p2, p3, cor)
lista_faces = [face]
# constructeur test - luz
branco = CorRGB(1.0, 1.0, 1.0)
luz_posicao = Ponto3D(1.0, 0.0, 2.0)
luz = Luz(luz_posicao, branco, branco, branco)
lista_luzes = [luz]
# constructeur test - camara
camara_posicao = Ponto3D(0.0, 0.0, 2.0)
olhar_para = Ponto3D(0.0, 0.0, 0.0)
vertical = Vetor3D(0.0, 1.0, 0.0)
distancia_olho_plano_projecao = 1.5
largura_retangulo_projecao = 2.0
altura_retangulo_projecao = 2.0
resolucao_horizontal = 50
resolucao_vertical = 50
camara = Camara(camara_posicao,
olhar_para,
vertical,
distancia_olho_plano_projecao,
largura_retangulo_projecao,
altura_retangulo_projecao,
resolucao_horizontal,
resolucao_vertical)
# constructeur test - cor de fundo
cor_fundo = CorRGB(0.0, 0.0, 0.2)
# constructeur test - ray tracer
ray_tracer = RayTracer(lista_faces, lista_luzes, camara, cor_fundo)
# teste a __str__
print(ray_tracer)
# constructeur test - renderiza
imagem = ray_tracer.renderiza()
imagem.guardar_como_ppm("teste1.ppm")
# referency test
# miss camara definition
# miss lista de faces definition
# miss cor de fundo definition
# miss lista de luzes definition
# ray tracer
ray_tracer = RayTracer(lista_faces, lista_luzes, camara, cor_fundo)
# renderization
imagem = ray_tracer.renderiza()
# file with renderization
imagem.guardar_como_ppm("teste2.ppm")
```

Table 1. Code to drive students to create the final class: Ray Tracer

METHODOLOGY

This research is a design research (Reeves, Herrington, & Oliver, 2005). The research question is: Is it possible to create an innovative multidisciplinary course joining algebra and a programming language that is positively evaluated by students and teachers? This is by itself a significant educational problem, but also it is made using a real project wich makes it pedagogicaly even more relevant.

This approach has been taught for seven years, reaching around 700 students. In the last two of those years an anonymous survey was presented to the students to get their feedback from the course. The approval rates of the last three years were monitorized. Iterative teachers' and researcher's reflection lead to changes and corrections to the project until to arrive to this stable status. For example, in the beginning teachers give the correct programing code to students before all Python classes, but now students program the code (following a guide) by themselves mostly in class with teacher's support. In the beginning the program was slower, it was improved to became faster, however, is still slow; in the following semester, we will study the implementation of a slightly different program wich is not a Ray Tracer but a Rendering Pipeline which produces images with lower quality but much faster.

DATA ANALYSIS

In 2016, the survey was compound of three questions:

- General evaluation of the course (0 to 20).
- Positive aspects of the course, to maintain.
- Negative aspects of the course, to change.

It was presented to the 45 students who completed the project immediately after getting their final mark and was answered anonimously. The mean grade given to the course was 15,5. And 35 over 45 gave 15 or more as the grade to the course. Many different aspects were approached as positive and negative. The most relevant to this research was:

- 14 students refer the interest of the course and final project, for example: "extremely positive the connection between mathematics and Python"; "final project strongly interesting"; "positive: connect mathematics and Python", "interesting subject"; "abstract subject that makes connection to reality".
- 9 students refer as negative that need more classes/support in Python while 14 refer that there is a high/enought support from teachers.
- 9 students had nothing to refer as negative while only two had nothing to refer as positive.

In 2017, the anonymous survey was presented online to all subscribed students and it was answered by 43 over the 113 subscribed. Beyond the questions of the previous semester some were added. One of them allows us to know that 72% of respondents were approved students, which as in the previous survey introduces some biase on results.

The mean grade given to the course was 16,1. And 17 over 31 gave 15 or more as the grade to the course. Many different aspects were approached as positive and negative. The most relevant to this research was: As positive: "The fact that mathematics complements programming and vice versa"; "Programming certain functions helped me to better understand Algebra"; "The creation of RayTracer, I found interesting and a good application to mathematics", "The project"; "I like the work of Python and Mathematics"; "the interconnection between the given subject and the final work is excellent and helps to consolidate all the knowledge and to test it."; "There should be MCG2"; "increases motivation";… As negtive: little time to finish the project (5 students reffered it).

All the four teachers of the course unanimously evaluate that approach as highly positive and that showing an immediate utilization of Algebra, using programming languages to create an image makes mathematics more interesting and engaging to those students. Moreover, Graphics computing, it's not just an example, is a central issue on a Multimedia Degree. Aldo, it creates a course with double difficulty since students must learn mathematics and programming. In 2015, the approval rate over assessed students was of 24/84=29% and over subscribed was 36/109=33%. In 2016, the approval rate over assessed students was of 36/72=50% and over subscribed was

36/110=33%. In 2017, the approval rate over assessed students was of 49/93=53% and over subscribed was 49/113=43%.

The approved students in fact learned all the parts since they grade more than 50%: in mathematic's final exam; in online individual (randomly generated) online homework (around 6 a semester) and also since they created and defended on a 30 minutes individual interview their project of object oriented programing in Python.

CONCLUSIONS AND FUTURE WORK

As conclusion, it is, in fact, possible to create a real multidisciplinary approach to teach algebra and a programming language together. Teachers and many students mostly found as positive that innovative approach that allows to experience the connection between mathematics and Python. However, some problems occurred, like mixing two difficult contents that makes the approval rate of students to be lower than desirable.

The Ray tracer program is a bit slow when renderizing the image, so we are studying the hypotesis of migrating to a Pipeline project that allows the image to be created faster but with lower quality.

REFERENCES

Artigue, M. (2016, March). Mathematics Education Research at University Level: Achievements and Challenges. In *First conference of International Network for Didactic Research in University Mathematics*.

Fangohr, H. (2004). A comparison of C, MATLAB, and Python as teaching languages in engineering. In *International Conference on Computational Science* (pp. 1210-1217). Springer Berlin Heidelberg.

Mills, J. E., & Treagust, D. F. (2003). Engineering education—Is problem-based or project-based learning the answer. *Australasian journal of engineering education*, *3*(2), 2-16.

NCTM. (2000). Principles and standards for school mathematics, Reston, VA.

Radenski, Atanas. "Python First: A lab-based digital introduction to computer science." *ACM SIGCSE Bulletin* 38.3 (2006): 197-201.

Reeves, T. C., Herrington, J., & Oliver, R. (2005). Design research: A socially responsible approach to instructional technology research in higher education. *Journal of Computing in Higher Education*, *16*(2), 96.

Schliep, A., & Hochstättler, W. (2002). Developing gato and catbox with python: Teaching graph algorithms through visualization and experimentation. In *Multimedia Tools for Communicating Mathematics* (pp. 291-309). Springer Berlin Heidelberg.

Teixeira, P., Matos, J. M., & Domingos, A. (2015). MATHEMATICS TEACHERS'INSTRUMENTAL GENESIS OF TECHNOLOGICAL MATERIALS. In 12th International Conference on Technology in Mathematics Teaching (p. 328).

Thomas, J. W. (2000). A review of research on project-based learning.