Holography Without Photography

Thad G. Walker

Department of Physics, University of Wisconsin-Madison,

Madison, Wisconsin 53706

September 21, 1998

Abstract

I detail a method for producing high quality Fourier transform holograms using a computer, laser printer, and overhead transparency film. The method should be useful for lecture demonstration and holography laboratories.

Given an arbitrary holographic phase/amplitude plate, computer-generated holography attempts to find a suitable binary encoding of the (complex) transmission function of the hologram. Several solutions to this problem have been described in this journal with applications to physics laboratories[1, 2, 3, 4, 5, 6]. The instructional use of computer generated holograms has the potential to enhance students' understanding of the holographic reconstruction process. In previous work, poor resolution of available computer output devices required the holograms to be photographically reduced for use in a laboratory optical system. Photographic processing limits the rate at which holograms can be produced to a few per hour at best, and can be unreliable in the hands of a novice, often limiting the learning experience for students attempting holography.

With the advent of low-cost, high resolution (> 300 dpi) laser printers, 75×75 element holograms can be produced directly on a $1'' \times 1''$ area of overhead transparency film. The entire process, which includes graphical description of an object, calculation of the hologram, and output to the printer, takes less than 5 minutes. The images reconstructed with a very simple optical rail are of quite high quality. With rapid cycle time and excellent results, such computer generated holograms should complement all-optical techniques for teaching students about holography. The procedure is conceptually very similar to that recently used in the demonstration of atom holography[7]. A related, though more complex, method using xerographic reduction at lower resolving power was described some time ago in the engineering literature[8].

Although many different types of holograms could be produced by variations on the method described here, Fourier transform holography is particularly simple to demonstrate. As its name implies, a Fourier transform hologram is the Fourier transform (with both phase and amplitude information) of the object to be reconstructed. When the hologram is placed in front of a lens and illuminated by a plane wave of light, the Fraunhofer diffraction pattern that appears in the focal plane of the lens is the Fourier transform of the hologram's transmission function[9], and therefore reconstructs the object. Since the Fourier transform of the object is in general complex, a Fourier transform hologram needs to modify both the amplitude and the phase of the transmitted light. Using lithographic and/or photographic techniques, for example, one can generate an arbitrary phase/amplitude hologram and therefore reconstruct an artificially produced object.

The central problem of computer-generated holography [10] is to simulate the Fourier transform hologram via a binary representation, in this case using only dark or transparent pixels output by a laser printer. The binary representation would work best if the hologram were real, positive, and had uniform modulation depth. Describing the image to be produced in terms of pixels at positions $\mathbf{r}_j$ with real amplitudes $a(\mathbf{r}_j)$ and phases ϕ_j, the ideal hologram would have the complex transmission function

$$A(\mathbf{r}') = \sum_j a(\mathbf{r}_j) e^{2\pi i \mathbf{r}' \cdot \mathbf{r}_j / f\lambda + i\phi_j} \tag{1}$$

This clearly cannot be well-represented by a binary function. The simplest way to change this into a real, positive function, is to take the real part of Eq. 1 and add a constant. Since the Fourier transform of a real even function is symmetric, this is equivalent to symmetrizing the object. Adding a constant to the transform adds a central bright spot to the reconstruction. Thus a real-valued hologram can reconstruct the intensity pattern of an arbitrary object at the price of adding a second, symmetric partner and a central bright spot.

The phases ϕ_j are arbitrary, since only the intensity and not the phase of the reconstruction is observed. This arbitrariness plays an important role in determining the image quality of the computer-generated hologram. Converting a positive, real hologram to binary is most easily done by selecting a threshold t and making a dark pixel at $\mathbf{r}'$ if $A(\mathbf{r}') < t$, with a transparent pixel otherwise. Since the binary approximation to the hologram does not distinguish values of $A(\mathbf{r}')$ that are much greater than t from those that are only slightly larger than t, it is important that the modulation depth of the hologram be as uniform as possible, *i.e.* that the Fourier transform of the reconstruction be spread as evenly as possible over many different pixels in the hologram. This is accomplished by choosing the phases ϕ_j

to be random, adding "white noise" to the reconstruction. The effect of the random phases is illustrated for a 1-D transform in Fig. 1.

The apparatus, as implemented at the University of Wisconsin, is depicted in Fig. 2. It consists of: 1) a computer with a graphics program and a C compiler; 2) a 600 dpi laser printer; and 3) an optical rail system consisting of a 5 mW HeNe laser, a 20X microscope objective, a 6″ focal length lens, and a mount for holding 35 mm slides. The two lenses form a telescope and focus the laser beam roughly 4 m from the hologram. With a mirror to fold the image back close to the hologram, the long distance makes the reconstructed object easily viewable by the student without further magnification.

Using the graphics program (Adobe Photoshop, GraphicConverter, and LView were all used with success), the object is described as an $R \times R$ pixel array. The value of R is the theoretical resolving power of the hologram, and is discussed below. To keep the C program as simple as possible, it is essential to use a straightforward graphics format. The "RAW" and "PGM" formats supported by most graphics programs are particularly nice, consisting simply of a header followed by a sequence of 8-bit words in binary format, each word describing a single pixel on the object. The C program, shown in Fig. 3, reads in the graphics file and calculates the hologram as

$$A(\mathbf{r}') = \sum_j a_j \cos(2\pi \mathbf{r}' \cdot \mathbf{r}_j / f\lambda + \phi_j) \tag{2}$$

where $\mathbf{r}_j$ is the position of the jth pixel on the object, $\mathbf{r}'$ is the position on the hologram, f is the focal length of the (compound) lens to be used for the reconstruction, λ is the laser wavelength, and ϕ_i is a random phase. The binary representation of the hologram is made by sampling $A(\mathbf{r}')$ at $4R \times 4R$ pixels on the hologram and finding the maximum and minimum values of A. All points for which $(A(\mathbf{r}') - A_{\min})/(A_{\max} - A_{\min}) > t$ are set to 1, all others to 0. The value $t = 0.6$ used for the holograms displayed here was determined by trial and error. Use of an FFT routine would considerably speed up the calculation of the hologram, but as it stands the calculation takes only 30 sec on the 200 MHz computer. The C program stores the binary representation to an output file which is printed by the graphics program to the laser printer (examples tested include the HP LaserJet 4000, HP LaserJet5L, and Apple LaserWriter Pro) onto overhead transparency film at 300 dpi resolution. The transparency is trimmed to fit into the 35 mm slide holder. A hologram so generated is shown in Figure 4.

The reconstruction optics are particularly simple. The lenses for the telescope were selected to make the laser beam roughly fill the hologram. A mirror 2m away reflects the light back to a screen next to the hologram for easy viewing. A photograph of a reconstruction is shown in Fig. 4. The two primary images are easily seen; much dimmer but also recognizable are supplementary higher-order images due to sampling and the discrete Fourier transform. A speckle pattern that arises from the relatively poor optical quality of the transparency film is also observed.

The resolving power of the hologram can be understood with reference to Fig. 4. The discrete Fourier transform repeats at intervals $f\lambda/2d$, where d is the pixel spacing on the hologram (this is an example of Nyquists theorem: the maximum spatial frequency that can be represented by samples a distance d apart is $1/2d$). Using diffraction grating terminology, the pixel spacing of d produces diffraction orders centered at integer multiples of $f\lambda/2d$. Each diffraction order has an associated reconstructed image and its symmetric partner. In order for the zeroth-order reconstruction to not overlap the first-order symmetric partner, the image must be limited to lie between the optical axis and $f\lambda/4d$. The diffraction-limited spot size is approximately $f\lambda/D$ where D is the diameter of the hologram. Thus the resolving power is $R \approx D/4d = 75$ for a $1'' \times 1''$ hologram at 300-dpi printer resolution. The observed resolving power of about 60 is close to this limit, and is probably limited by aberrations in the lens system.

Comparable results should be attainable using other computers, software packages, and laser printers, though poor results were obtained when the printers were used at their advertised resolution. Thus, for 300 dpi resolution it is probably best to use a 600 dpi laser printer.

The example given here is a simple Fourier transform hologram. Using the computer, it is easy to make other types as well. As an example, one can add zone lenses to the hologram; a simple way to do this is to make

$$A(\mathbf{r}') = \sum_i \cos(2\pi|\mathbf{r}' - \mathbf{r}_i|^2/f\lambda + \phi_i) \tag{3}$$

This has the property that the two primary images and the laser spot image in different planes; this adds background to the image, but allows the resolving power to be increased to $D/2d$. Another interesting exercise is to form an intermediate image by adding two more lenses to the setup; this allows spatial filtering of the hologram to remove the auxiliary images and the laser spot. Other possibilities include simulating axicons (linear variation of phase shift with radial position, as opposed to a lens which has a quadratic variation), or 3-D holograms produced by allowing the zone plates for different object points to have different focal lengths.

In summary, it is practical to generate holograms with readily available and inexpensive computer equipment, avoiding photographic processing. The methods should be applicable to lecture demonstration, as a supplement to all-optical holography laboratories, or as a stand-alone holography experiment.

Support came from the NSF and the Packard Foundation. I thank J. Hanesworth for help with the computer graphics.

References

[1] J. S. Marsh and R. C. Smith, “Computer holograms with a desk-top calculator”, Am. J. Phys. **44**, 774-777 (1976).

[2] C. W. Leming and O. P. Hastings III, “Computer-generated microwave holograms”, Am. J. Phys. **48**, 938-939 (1980).

[3] J. S. Marsh, “Optical scanner for the Apple computer”, Am. J. Phys. **53**, 792 (1985).

[4] Xiangxi-Chen, J. Huang, and E. Loh, “Computer-assisted teaching of optics”, Am. J. Phys. **55**, 1129-1133 (1987).

[5] A. E. Macgregor, “Computer generated holograms from dot matrix and laser printers”, Am. J. Phys. **60**, 839-846 (1992).

[6] S. Trester, “Computer simulated holography and computer generated holograms”, Am. J. Phys. **64**, 472-476 (1996).

[7] M. Morinaga, M. Yasuda, T. Kishimoto, and F. Shimizu, “Holographic manipulation of a cold atomic beam”, Phys. Rev. Lett. **77**, 802-805 (1996).

[8] R. A. Gonsalves and J. D. Prohaska, “Simple laboratory experiment on computer-generated holograms”, Proc. SPIE **938**, 472 (1988).

[9] E. Hecht, *Optics*, 2nd ed. (Reading, MA: Addison-Wesley Publishing Company, 1987), Ch. 11.

[10] W. Lee, “Computer-generated holograms: techniques and applications”, Progress in Optics **16**, 119-232 (1978).

[11] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C*, 2nd ed. (New York:Cambridge University Press, 1993).

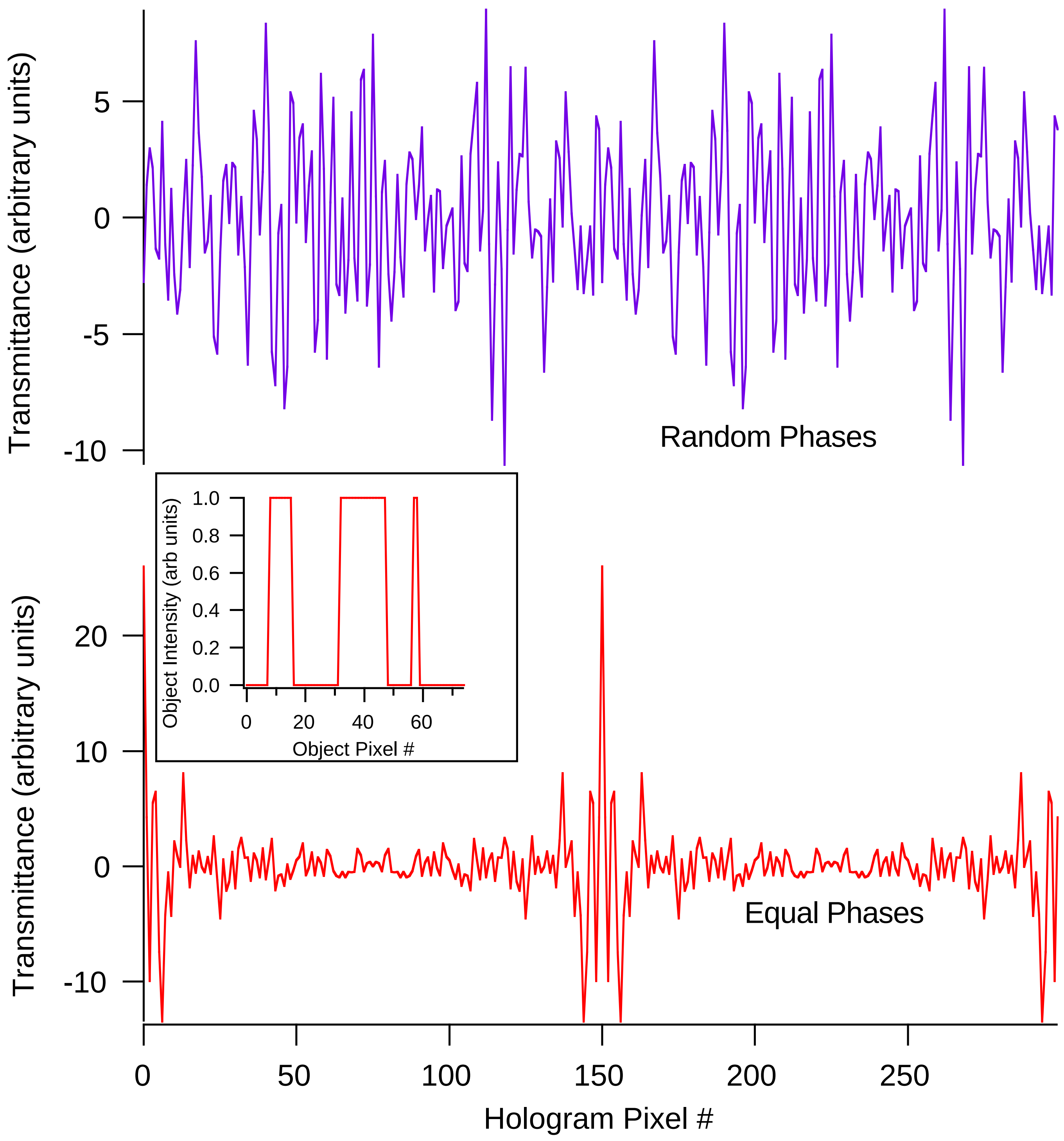
Figure Captions

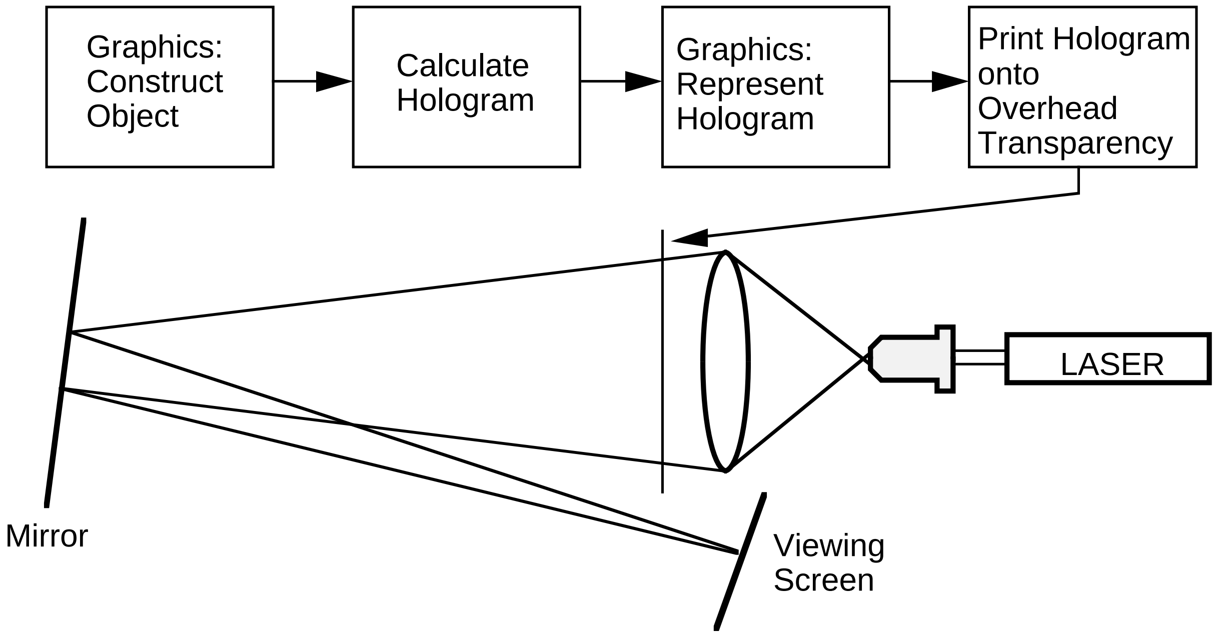
Figure 1: 1-D Fourier transform holograms with either random phases for the various object points, or uniform phase. The random phase transform spreads the information about the object over many pixels, while the uniform phase concentrates the object information onto a few pixels. A binary representation of random phase hologram therefore gives a better reconstructed image. Inset: 1-D object.

Figure 2: Schematic diagram for computer generated holography.

Figure 3: Computer program used to process the graphics files for the hologram. Memory management routines are from Numerical Recipes (Ref.[11]).

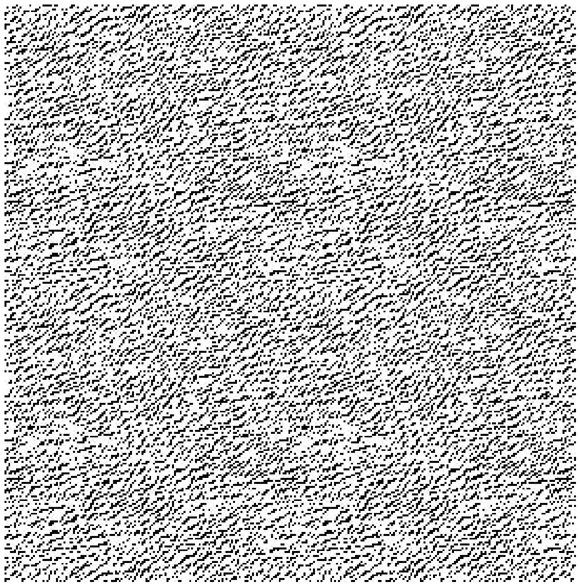
Figure 4: Fourier transform hologram and its reconstruction.

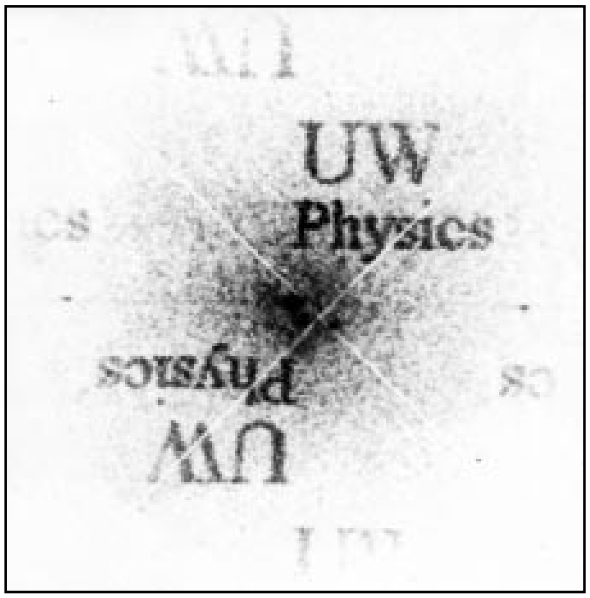

Transmittance (arbitrary units)
5
0
-5
-10
Random Phases
Object Intensity (arb units)
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
Object Pixel #
Transmittance (arbitrary units)
20
10
0
-10
Equal Phases
0
50
100
150
200
250
Hologram Pixel #


Graphics: Construct Object
Calculate Hologram
Graphics: Represent Hologram
Print Hologram onto Overhead Transparency
LASER
Mirror
Viewing Screen

```
//holo.c--simple program for computer-generated holography
//Thad G. Walker University of Wisconsin-Madison
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include "nrutil.h" //Numerical Recipes routines for memory allocation
#define twopi 6.283185307
#define R 75        //Resolving power of hologram
#define N 300       //hologram width in pixels
#define thres 0.6   //threshold for binary approximation

 void main(){
    char pixel,obj;
    int X,Y,row,col,junk;
    float **pix,kdf=twopi/2.0,ranphi,max=0.0,min=0.0,res=R;
    FILE *out,*in;
    out=fopen("holo.pgm","w");      // output file, N X N size graphics file
    in=fopen("uwphysics.pgm","r");  //input file, R X R size graphics file
    fprintf(out,"P5 %d %d 255\n",N,N);//header
    fscanf(in,"%1s %1s %d %d %d",&junk,&junk,&junk,&junk,&junk);//header
    srand(17);                      //seed random number generator
    pix=matrix(0,N-1,0,N-1);        //allocate memory for graphical output
    for(X=0;X<N;X++){for(Y=0;Y<N;Y++){pix[X][Y]=0.0;}}      //initialization
    //construct real hologram from input image
    for(row=0;row<R;row++){for(col=0;col<R;col++){
        fscanf(in,"%1c",&obj);      //read 1 byte at a time
        if(obj!=0){                 //all non-zero bytes given same intensity
            ranphi=((float) rand())/((float) RAND_MAX)*twopi;   //select random phase
            for(X=0;X<N;X++){for(Y=0;Y<N;Y++){
                pix[X][Y]+=cos(kdf*(X*(1.-row/res)+Y*col/res)+ranphi);//add to hologram
            }}
        }
    }}
    //find max min
    for(X=0;X<N;X++){for(Y=0;Y<N;Y++){
        if(pix[X][Y]>max)max=pix[X][Y];
        if(pix[X][Y]<min)min=pix[X][Y];
    }}
    //print output file
    for(X=0;X<N;X++){for(Y=0;Y<N;Y++){
        if((pix[X][Y]-min)/(max-min)>thres){pixel=255;
        }else{pixel=0;}
        fprintf(out,"%c",pixel);
    }}
}
```




UW
Physics